Grid Computing para la recuperación de datos climáticos

Sergio Nesmachnow, Sebastián García, Gabriel Usera
Univ. de la República, Uruguay
{sergion,sgarcia,gusera}@fing.edu.uy

Francisco Brasileiro
Univ. Federal de Campina Grande, Brasil
fubica@dsc.ufcg.edu.br

Abstract

Este artículo presenta los avances en el proyecto Digi-Clima Grid, que propone la utilización de técnicas de computación distribuida sobre plataformas cluster y grid para la digitalización y recuperación de registros pluviográficos del acervo histórico climático de Uruguay. Se presentan implementaciones eficientes del problema de digitalización utilizando técnicas de computación distribuida sobre plataformas cluster y grid, y una evaluación experimental que ejemplifica los beneficios de las soluciones diseñadas para el procesamiento automático de digitalización de registros pluviográficos.

1. Introduccción

El estudio del comportamiento de variables climáticas a lo largo del tiempo es de gran interés en diversas áreas de la ciencia, y también para la producción industrial y el manejo de catástrofes, entre otras aplicaciones. El impacto potencial de la predicción del comportamiento climático es muy importante a corto plazo para la toma de decisiones, por ejemplo en la producción agrícola, y también a largo plazo como ser para predecir si una isla se hundirá o no dentro de 100 años. Para ciertas aplicaciones, como el control de bombeos en el saneamiento de una ciudad, o la predicción del nivel de un río para advertir inundaciones, basta con conocer el estado actual de dichas variables o su pasado reciente. Sin embargo, existen muchos problemas para los que es necesario contar con registros correspondientes a largos períodos de tiempo, en términos climáticos, de 30 años en adelante.

Por estos motivos, la comunidad científica está interesada en recuperar datos climáticos recabados a través de los años. A medida que se retrocede en el tiempo, los registros son más escasos y más difíciles de recuperar y conservar.

En Uruguay, existen registros sistemáticos y extensivos de diversas variables climáticas, recabados desde hace más de 100 años. Los registros se encuentran almacenados en papel, lo cual amenaza su preservación y hace imposible su utilización sin pasar por una etapa previa de transcripción.

Estos registros contienen información en lenguaje natural en algunos casos. En otros, como ser los registros pluviográficos, es posible automatizar el procedimiento de transcripción mediante una digitalización.

Este es el caso que con el que trata el proyecto Digi-Clima Grid: desarrollar una aplicación eficiente capaz de digitalizar la salida escaneada de un pluviógrafo y almacenarla en una base de datos de registros histórico-climáticos.

Ejecutando en un computador estándar, se requieren unos 10-15 minutos para poder procesar una imagen correspondiente a un día de registro pluviográfico. Considerando que se dispone de más de un siglo de registros pluviográficos para estaciones meteorológicas en varios puntos del país, es necesario aplicar técnicas de computación de alto desempeño para poder obtener los resultados del procesamiento en tiempos razonables.

El proyecto Digi-Clima Grid [3] propone aplicar técnicas de computación distribuida para el procesamiento eficiente del proceso de digitalización, utilizando infraestructuras *cluster*— el *Cluster FING* de la Facultad de Ingeniería, Universidad de la República, Uruguay [2]—y plataformas *grid* como las del proyecto de grid latinoamericana-europea *GISELA* [6] y el middleware para grid/cloud OurGrid [1].

El proyecto involucra a los grupos de investigación en Mecánica de los Fluidos Computacional, Hidrología Superficial y al Centro de Cálculo, de la Facultad de Ingeniería, Universidad de la República, Uruguay; y al Departamento de Sistemas e Computação, de la Universidade Federal de Campina Grande, Brasil. El proyecto ha recibido apoyo económico de la Organización Meteorológica Mundial; de la Dirección Nacional de Aguas y de la Intendencia Municipal de Montevideo, Uruguay; y de la Federación de Investigación en Tecnologías de la Información y la Comunicación para América Latina y el Caribe.

Las principales contribuciones de este trabajo consisten en la descripción de la aplicación de técnicas de computación distribuida para el desarrollo de una aplicación eficiente para el procesamiento y digitalización de registros pluviográficos nacionales, capaz de ejecutar sobre plataformas de cómputo cluster y grid.

La estructura del documento se describe a continuación.

La Sección 2 describe el proceso de digitalización de registros histórico-climaticos, la aplicación Digi-Clima y reseña trabajos relacionados. Los algoritmos diseñados en el marco del proyecto se presentan en la Sección 3, conjuntamente con los detalles de su ejecución en las distintas plataformas de cómputo consideradas. La Sección 4 reporta los resultados experimentales de la validación de los algoritmos distribuidos diseñados. Por último, la Sección 5 presenta las conclusiones y principales líneas de trabajo actual y futuro.

2. Digitalización de registros histórico-climaticos

Esta sección describe el problema de digitalización de registros histórico-climáticos, introduce la aplicación Digi-Clima y presenta un relevamiento de trabajos relacionados.

2.1. La aplicación Digi-Clima

En Uruguay, la Dirección Nacional de Meteorología lleva registros sistemáticos y extensivos de variables climáticas desde principios del siglo pasado, incluyendo tablas de datos manuscritas, reportes descriptivos y también registros gráficos de pluviógrafos y otros aparatos. Estos datos son de gran valor para la comprensión de fenómenos climáticos y para la predicción climática con modelos numéricos. Sin embargo, resulta muy difícil analizarlos al encontrarse registrados en papel. Además, su preservación está en riesgo por el deterioro inherente del material utilizado.

Para garantizar la preservación de estos registros histórico-climáticos, el proyecto Digi-Clima fue concebido para proveer un medio semi automático para la digitalizar los registros históricos de intensidad de lluvia, haciendo posible la preservación de los datos y su utilización.

Los registros de intensidad de lluvia consisten en cintas de papel milimetrado (*bandas*), en las que un pluviógrafo registró en cierto período de tiempo la cantidad de lluvia acumulada. Cada banda contiene en una cara el registro del pluviógrafo y en la otra anotaciones manuscritas que indican las fechas de inicio y fin del registro así como el registro del pluviómetro para ese rango, entre otros datos.

Los datos del pluviógrafo consisten en una gráfica continua y monótona creciente en tramos, representando la acumulación de lluvia hasta alcanzar la capacidad del aparato. Luego sigue una caída en vertical hasta el cero y se vuelve a empezar. La Figura 1 presenta un ejemplo de registro pluviográfico para la ciudad de Artigas en el año 1980.

En la banda de ejemplo se aprecian algunos de los problemas asociados a su digitalización, ocasionados por defectos del propio instrumento y de la medición realizada (línea discontinua, manchas de tinta, diferentes niveles de intensidad en la línea de registro), pero también por la presencia de anotaciones manuscritas hechas al retirar la cinta.

La *aplicación Digi-Clima* consiste en un programa Matlab desarrollado para extraer el histórico de lluvia acumulada a partir de una imagen como la de la Figura 1, utilizando técnicas de segmentación y separación cromática, seguimiento de curvas, conteo e interpolación. La aplicación hace uso de herramientas y toolboxes específicos de Matlab para procesamiento de imágenes e interpolación, y existe una versión en desarrollo en Octave. Un diagrama del procesamiento de Digi-Clima se presenta en la Figura 2.

Asimismo, los datos manuscritos registrados al dorso de la banda, que contienen las fechas de comienzo y fin del registro, y la medición del pluviómetro para ese período (total de lluvia acumulada), son esenciales para catalogar el registro pluviográfico. Para la extracción de estos datos se diseñó una aplicación que simplifica la transcripción manual de los datos por parte de un operador.

Las imágenes correspondientes a ambas caras de las bandas se digitalizaron utilizando scanners y se almacenaron en formato JPEG. La primer etapa del proyecto involucró la digitalización de 8000 bandas correspondientes a las ciudades de Montevideo, Rocha, Paysandú y Artigas, en Uruguay, tomados entre 1968 y 2010. Actualmente se cuenta con más de 30.000 imágenes de diversas estaciones meteorológicas, y nuevos datos son requeridos por la Dirección Nacional de Meteorología periódicamente.

La aplicación Digi-Clima tarda unos 10-15 minutos en procesar una única imagen. El tiempo de ejecución secuencial para las imágenes de la etapa inicial hubiera demandado más de dos meses de cómputo en un computador estándar. Por este motivo, y considerándose que se estima que el registro nacional histórico cuenta con más de 50.000 bandas, el proyecto Digi-Clima Grid propuso la utilización de técnicas de computación de alto desempeño para distribuir el procesamiento y resolver eficientemente el problema de digitalización de registros histórico-climáticos.

2.2. Trabajos relacionados

Existen varios proyectos previos que han abordado problemas similares al que afronta Digi-Clima.

Una de las primeras iniciativas que enfrentó el problema fue el proyecto CLIWOC [5], que construyó una base de datos a partir de cientos de miles de observaciones climáticas tomadas de bitácoras de barcos holandeses, británicos, españoles y franceses entre 1750 y 1850.

El trabajo del proyecto CLIWOC se dividió en equipos que se especializaron en el análisis de bitácoras, transcribiendo fecha, ubicación geográfica, estado del tiempo, dirección y fuerza del viento, estado del mar, reportes de hielo, y temperatura y presión del aire. El proyecto CLIWOC también construyó un diccionario que permitió unificar los términos utilizados en las bitácoras, en varios idiomas, a lo largo de 100 años.

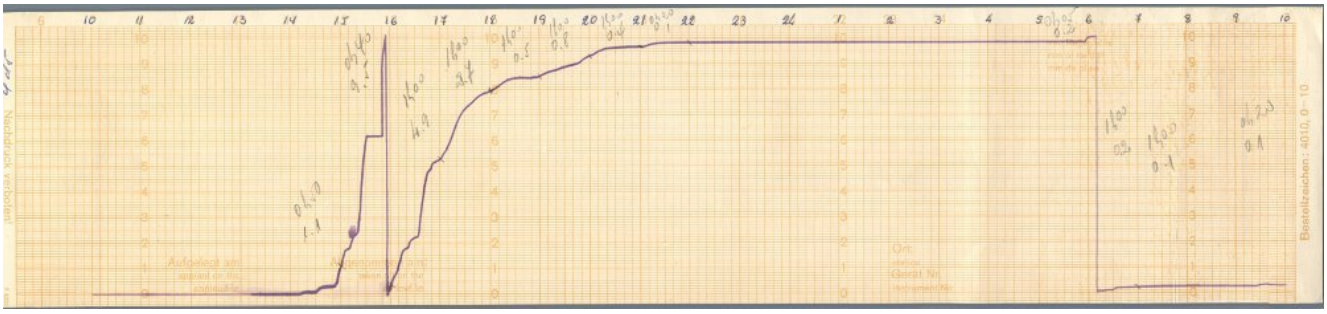

Figura 1. Ejemplo de banda pluviográfica (ciudad de Artigas, Uruguay, año 1980)

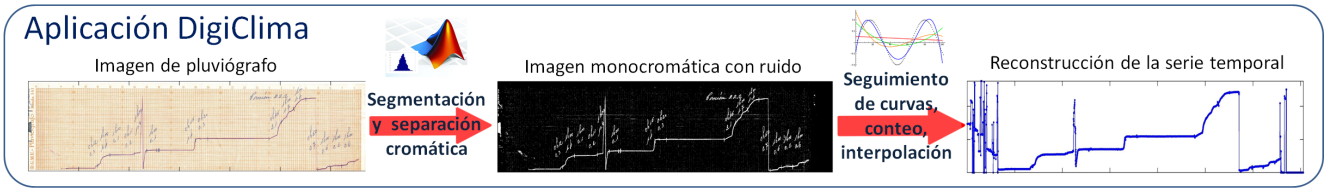

Figura 2. Diagrama de procesamiento de la aplicación Digi-Clima

De interés para el proyecto Digi-Clima, se destacan los controles de calidad en la transcripción de datos, para los que se eligió al azar el 5 % de las transcripciones, se las revisó, y en caso de encontrar errores significativos se volvía a procesar las transcripciones del mismo período.

En la misma línea de investigación, el proyecto RECLAIM [10] (en desarrollo) propone continuar el trabajo de CLIWOC, procesando datos que se dejaron de lado en el proyecto original. RECLAIM se propone a largo plazo procesar registros de la era instrumental, desde 1853 a la fecha.

El proyecto Old Weather [9] se centra en la etapa de transcripción. Propone extraer datos a partir de imágenes digitales de los documentos, utilizando operadores humanos voluntarios. Para hacer la tarea más atractiva y atraer más colaboradores, desarrolló un juego online. Un jugador se suscribe y elige un barco, de los que se tiene registros a digitalizar. El usuario comienza con un rango bajo y a medida que transcribe registros, va obteniendo puntos que le permiten subir de rango. De este modo, el proyecto Old Weather obtiene digitadores gratuitos, permitiendo hacer múltiples transcripciones de cada registro, y obtener resultados de gran calidad. La disciplina que se ocupa de la resolución de problemas de cómputo por parte de operadores humanos, se conoce como "human computing" y el uso de videojuegos en esta área es una de las líneas de investigación más recientes del área. Su utilización se propone como trabajo futuro en el proyecto Digi-Clima.

Los tres proyectos comentados están relacionados con Digi-Clima. Ellos muestran que existe una motivación de la comunidad científica internacional para recuperar información sobre variables climáticas de registros históricos.

El relevamiento de los proyectos en cuestión permitió conceptualizar el estado del arte sobre técnicas de digitalización de manuscritos. Sin embargo, los proyectos relacionados no han propuesto soluciones eficientes para la automatización del proceso de digitalización, ya que se basan en técnicas de human computing. Asimismo, tampoco hay antecedentes de uso de computación distribuida o de alto desempeño para acelerar el proceso de recuperación de los datos. Por estos motivos, el proyecto Digi-Clima Grid aporta una contribución a la línea de investigación, al proponer el estudio de técnicas de computación distribuida para resolver el problema.

3. Computación de alto desempeño aplicada a Digi-Clima

Esta sección describe las soluciones implementadas para la ejecución eficiente de Digi-Clima sobre plataformas de computación de alto desempeño.

3.1. Modelo de paralelismo

Las estrategias desarrolladas toman a la aplicación Digi-Clima como caja negra y pretenden sacar provecho de diferentes infraestructuras de cómputo, que se caracterizan por ser sistemas de memoria distribuida.

El procesamiento de las imágenes digitalizadas es independiente, por lo cual es posible aplicar una estrategia de paralelismo sobre los datos de entrada, utilizando una partición en imágenes o conjuntos de imágenes.

La paralelización consiste en ejecutar múltiples instancias de Digi-Clima simultáneamente, para datos de entrada diferentes, sin existir comunicaciones entre tareas durante el procesamiento. Las comunicaciones serán únicamente para mantener información compartida sobre el estado de procesamiento de las imágenes del banco, evitando así el procesamiento de una imagen por parte de dos instancias diferentes. Como consecuencia, el único overhead que se tendrá al utilizar más instancias es el dado por la mutua exclusión del acceso a la información compartida sobre el estado de procesamiento de las imágenes y por el acceso al banco de imágenes. El acceso a la información compartida se dará en el peor de los al comenzar y finalizar el procesamiento de una imagen, pero se puede reducir haciendo que las instancias procesen conjuntos de varias imágenes. Entonces la estrategia propuesta permite teóricamente escalar con speedup lineal, siempre y cuando no se alcance el límite en el acceso al medio de almacenamiento para el banco de imágenes que ofrezca la infraestructura correspondiente.

El enfoque de caja negra permite concentrarse en el paralelismo de datos e independizarse de la aplicación en sí, permitiendo mejoras contínuas al mecanismo de procesamiento mediante versiones actualizadas de la aplicación, que se encuentra en constante desarrollo.

Existen dos opciones para ejecutar Digi-Clima: i) ejecución interpretada en el entorno y ii) ejecución de un binario compilado para una plataforma específica, utilizando un *runtime* de uso es libre (el Matlab Compiler Runtime, MCR) Respecto al paralelismo, puede utilizarse una ejecución de hilo único o una multihilo.

Las implementaciones paralelas presentadas en este trabajo permiten trabajar con versiones interpretadas o compiladas y de hilo único o multihilo. Sin embargo, se ha considerado apropiado trabajar con una implementación *compilada* y *de hilo único*.

Las diversas alternativas de aplicar paralelismo multihilo dentro de la aplicación Digi-Clima fueron descartadas luego de analizar empíricamente, utilizando profilers, sus pobres beneficios en cuanto a tiempo de ejecución, principalmente por la existencia de secciones seriales en el procesamiento. Además, estas alternativas implicarían modificar el código de la aplicación a ejecutar, con las consiguientes complicaciones inherentes al problema de tratamiento de imágenes considerado, la implementación de funciones externas en los toolboxes de Matlab, la gestión de los datos intermedios generados en cada etapa de procesamiento, etc.

El objetivo del proyecto no consiste es optimizar el procesamiento de *una* imagen, sino el procesamiento de un gran número de imágenes, por lo cual adoptar el enfoque de memoria distribuida es la decisión más adecuada.

A continuación se presentan las implementaciones desarrolladas en este proyecto para ejecutar Digi-Clima en infraestructuras de cómputo de memoria distribuida.

3.2. Paralelismo en cluster

En esta sección se presentan las implementaciones desarrolladas para ejecutar Digi-Clima en infraestructuras cluster: i) una implementación del modelo maestro-esclavo con MPI, utilizando asignación dinámica de tareas, y ii) una implementación basada en la ejecución de trabajos independientes, que interactúan con un servidor de base de datos.

3.2.1. Maestro-esclavo con MPI

Esta solución se basa en el modelo maestro-esclavo con partición de dominio, utilizando la biblioteca MPI [7] para la sincronización y comunicación entre procesos.

Se trabaja considerando la aplicación como caja negra, y la unidad del dominio son las imágenes. El proceso maestro se ocupa de asignar imágenes a los esclavos para procesar, y los procesos esclavos ejecutan la aplicación Digi-Clima para cada imagen asignada. El tiempo de ejecución de Digi-Clima depende de cada imagen, y no existe una forma sencilla de estimarlo. Por lo tanto, para evitar los desbalances de carga que surgirían en caso de trabajar con una partición estática del dominio, se decidió utilizar conjuntos de pocas imágenes y aplicar una asignación dinámica de tareas para balancear la carga entre los esclavos. A cada esclavo se le asigna un conjunto inicial de imágenes, y luego se les asignan nuevas a demanda. Cuando finaliza su procesamiento, cada esclavo requiere nuevas imágenes al maestro

En la Figura 3 puede verse un esquema básico de la arquitectura del modelo maestro-esclavo con MPI.

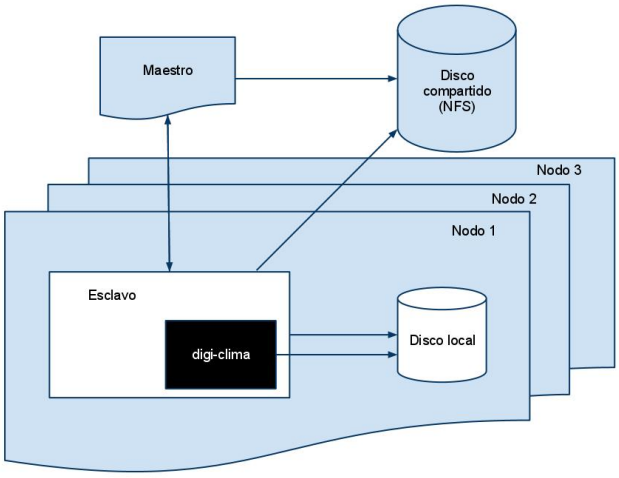

Figura 3. Arquitectura maestro-esclavo MPI

Un mecanismo simple de tolerancia a fallos fue implementado para no perder la información procesada en caso de que ocurra un evento inesperado que impida continuar con la ejecución. El proceso maestro mantiene y actualiza

en un archivo metadatos básicos de las imágenes, incluyendo su ruta en disco y su estado de procesamiento, leyéndolos en la etapa de inicialización y permitiendo reanudar el procesamiento donde la ejecución anterior lo dejó.

Para la implementación de esta solución se utilizó el lenguaje C y la biblioteca MPI para el código del maestro y los esclavos. Además, se desarrolló un conjunto de scripts bash para ejecución y utilitarios: i) descriptor del trabajo para su ejecución, ii) inicialización del entorno de ejecución en los esclavos, iii) inicialización del proceso maestro y tolerancia a fallos (recorrer archivos de salida y chequear metadatos) y iv) scripts para renombrar las imágenes de modo que tengan nombres numéricos secuenciales.

3.2.2. Trabajos independientes con base de datos

Esta solución consiste en la ejecución de procesos independientes (*pilot jobs*) y utilizar una base de datos para llevar los metadatos de las imágenes.

El pilot job es un proceso implementado en Python que itera los siguientes pasos: i) obtener la ruta de una imagen sin procesar de la base de datos, e indicar en esta que está siendo procesada; ii) ejecutar Digi-Clima para dicha imagen; y iii) registrar en la base de datos que la imagen fue procesada. El procedimiento se repite hasta que todas las imágenes han sido procesadas. La Figura 4 presenta un esquema del flujo de trabajo del un pilot job.

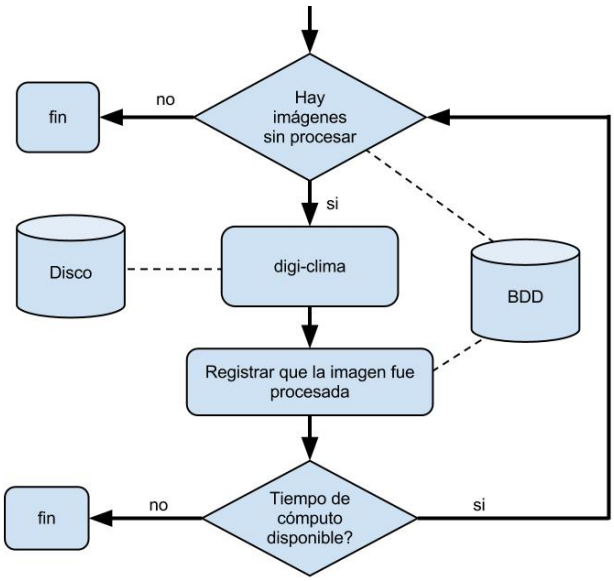

Figura 4. Diagrama del pilot job en cluster

Dado que se requiere un tiempo considerable para procesar una imagen, las transacciones que el pilot job realiza sobre la base de datos son espaciadas en el tiempo. Además, las transacciones son sencillas, permitiendo escalar a un gran número de pilot jobs simultáneos.

Se utilizó el manejador de bases de datos MySQL por su versatilidad y por ser de código libre, con un API estable del cliente en varios lenguajes de programación, incluyendo los utilizados en el proyecto: C, Matlab y Python.

El diseño utilizando un pilot job por procesador y Digi-Clima compilado para utilizar un único hilo de procesamiento, resuelve el balance de cargas. Mientras existan imágenes para procesar, el pilot job estará haciendo uso total del procesador asignado, y cuando finalice con su procesamiento, el pilot job deja el procesador a disposición del gestor del cluster. La solución también ofrece cierto nivel de tolerancia a fallos, al registrar las imágenes procesadas en la base de datos. La caída de uno de los pilot jobs sólo ocasiona la pérdida del procesamiento de la imagen en curso y no de las anteriores.

Se tienen luego puntos de falla centralizados, por un lado en el sistema de archivos utilizado tanto para la lectura de las imágenes como para el almacenamiento de los resultados, y por otro en el servicio de la base de datos. Para resolver dichos puntos de falla centralizados, podría utilizarse cualquier sistema de tolerancia a fallas en el sistema de archivos, así como en el servicio de MySQL, para los que existen diversas soluciones que podrían aplicarse de forma transparente a la aplicación propuesta.

3.3. Pilot jobs en grid GISELA

Esta subsección presenta la solución diseñada para aprovechar la capacidad de cómputo en infraestructuras grid para ejecutar la aplicación Digi-Clima. La solución implementada está basada en la infraestructura y herramientas disponibles en el grid de GISELA.

Los principales aspectos a resolver para la ejecución de Digi-Clima en una infraestructura grid son: i) distribuir las imágenes en el grid; ii) evitar el procesamiento de la misma imagen por parte de dos instancias de Digi-Clima simultáneamente; y iii) recuperar los resultados y mantener la referencia de a qué imagen corresponden.

Además, al tratase de una infraestructura grid, los recursos de cómputo se encuentran bajo diferentes dominios de administración, lo que puede impedir la instalación del runtime de Matlab en todos los nodos de cómputo (WNs). Para ejecutar Digi-Clima es necesario enviar además de la aplicación, el runtime de Matlab, que tiene un tamaño de 200MB. Como consecuencia, la inicialización de un WN para ejecutar Digi-Clima demanda un tiempo no despreciable.

La implementación propuesta se basa en pilot jobs, que una vez enviados y asignados a un elemento de cómputo, entran en un bucle ejecutando Digi-Clima en cada iteración. El runtime de Matlab sólo se descargará una vez por WN y luego se utilizará para procesar múltiples imágenes, mitigando las demoras dadas por el envío del trabajo y por la preparación del entorno de ejecución.

Los servicios de archivos del grid se utilizan para distribuir las imágenes y para recuperar los resultados. El servicio de metadatos AMGA [8] es utilizado para llevar los metadatos de las imágenes, incluyendo el estado de procesamiento y el mapeo entre el nombre original del archivo correspondiente a la imagen y los resultados correspondientes.

A continuación se presentan las soluciones implementadas para enviar las imágenes al grid y para procesarlas.

Script para subir imágenes al grid. Inicialmente el script crea un directorio en AMGA y agrega los atributos necesarios para los metadatos de las imágenes: i) el identificador de la imagen en el storage element (SE) del grid; ii) el estado de procesamiento de la imagen; iii) el nombre original del archivo de la imagen; iv) una marca de tiempo que indica la última actualización realizada sobre los metadatos de la imagen y v) el identificador en el grid del último trabajo en actualizar los metadatos de la imagen. A continuación, se utiliza el comando `lcg-cr` para enviar las imágenes al grid, interactuando con el servicio de catálogo de archivos LFC para dar de alta cada imagen por su nombre, pero también da de alta una réplica inicial en un SE. Una vez finalizada la ejecución del script, se tiene una copia del banco de imágenes en el sistema de archivos del grid, así como los metadatos inicializados en AMGA, por lo que se está en condiciones de ejecutar los pilot jobs.

Pilot job. El pilot job tiene dos componentes principales: i) el *archivo de descripción del trabajo* (JDL) para su envío al sistema de manejo de trabajos (WMS), y ii) el script que implementa el la ejecución de Digi-Clima en un WN. La Figura 5 presenta un diagrama de la ejecución del pilot job.

El JDL presenta la especificación del trabajo, incluyendo parámetros como el nombre del archivo a ejecutar y sus atributos, nombres para la entrada y salida estándar, nombres de los archivos que serán enviados junto con el trabajo al WMS, intentos de ejecución en caso de falla y requisitos de ejecución (por ejemplo, arquitectura del nodo).

Un esquema del script que implementa el pilot job se presenta en la Figura 6. El pilot job inicializa el entorno para poder trabajar con los comandos `lcg` (líneas 2-4). Luego, descarga los archivos necesarios para ejecutar (bucle en líneas 5-17), intentando para cada archivo localizar la mejor réplica para la descarga, y copiarlo al WN. Para ubicar la réplica más cercana se utiliza el comando `lcg-lr` (línea 8), que lista las réplicas existentes para un nombre lógico, buscando el SE por defecto para el WN en que se está ejecutando (línea 9). Si el SE por defecto no se encuentra definido en el WN o no existe una réplica del runtime, se intenta utilizar un SE en el que se sabe que hay una réplica (línea 14), o se usa la primera réplica que devuelve el catálogo (línea 17). Lego de elegir la réplica se procede a descargar el archivo, utilizando el comando `lcg-cp` (línea 18).

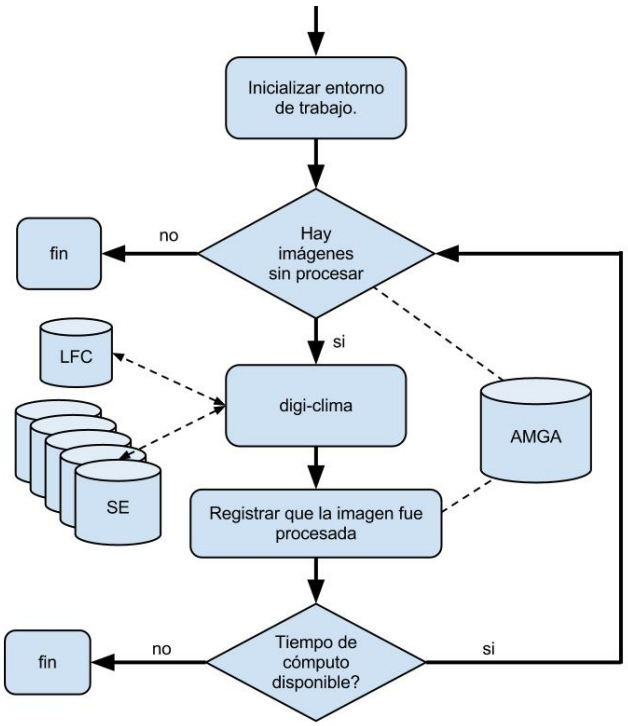

Figura 5. Diagrama del pilot job GISELA

Luego de descargar el runtime, las bibliotecas necesarias y el ejecutable de Digi-Clima, el pilot job iterativamente ejecuta el script `Digi-Clima.sh` (líneas 21-26). El pilot job finaliza cuando el valor de retorno de `Digi-Clima.sh` indica el fin de procesamiento de las imágenes las imágenes o cuando ocurre un error en el procesamiento.

Un esquema del script `digi-cima.sh` que ejecuta el pilot job en cada iteración se presenta en la Figura 7. Este script obtiene una imagen para procesar, ejecuta la aplicación Digi-Clima y marca la imagen como procesada.

El script `Digi-Clima.sh` inicialmente realiza una consulta SQL al servicio de metadatos AMGA para obtener una imagen sin procesar, mediante el comando `mdcli` (línea 2). Luego intenta actualizar el estado de la imagen obtenida para indicar que está siendo procesada, condicionando la actualización a que la imagen aún se encuentre sin asignar (líneas 5-6), ya que otro pilot job puede haber seleccionado la misma imagen en la ventana de tiempo entre la consulta y el intento de actualización de AMGA. Esta actualización condicional realizada como operación atómica sobre AMGA permite resolver la condición de carrera dada por dos pilot jobs que seleccionan la misma imagen, y sólo uno de ellos tendrá éxito en la actualización. Si la actualización falla, el pilot job continúa con su ciclo de procesamiento.

Cuando la actualización de estado de la imagen tiene éxito, se intenta obtener la imagen desde la réplica más cercana, del mismo modo que en el script bash del pilot job.

```
1  # Inicializar el entorno LFC y LCG
2  export vo=prod.vo.eu-eela.eu
3  export LCG_LOCATION=/opt/grid/ui/lcg
4  export LCG_GFAL_INFOSYS=bdii-eela.ceta-ciemat.es:2170
5  for each file do # Descargar achivos
6    SEL_REPL= file_uri
7    # Intentar obtener la mejor réplica
8    REPL=$(lcg-lr file_uri | grep "${$VO_PROD_VO_EU_EELA_EU_DEFAULT_SE}")
9    if [ "$REPL" != "" ]; then
10     # utilizar replica encontrada en DEFAULT_SE
11     SEL_REPL=$REPL
12   else
13     # Intentar utilizar "se01-tic.ciemat.es"
14     REPL=$(lcg-lr file_uri | grep "se01-tic.ciemat.es")
15     if [ "$REPL" != "" ]; then
16       # se01-tic.ciemat.es no disponible, intentar LFN
17       SEL_REPL=$REPL
18   lcg-cp -checksum -checksum-type md5 -verbose $SEL_REPL file_name
19 # Ejecutar Digi-Clima
20 END_EXECUTION=0
21 while [ $END_EXECUTION -eq 0 ] ; do
22   /bin/sh ${JOB_DIRECTORY}/Digi-clima.sh ${JOB_DIRECTORY}
23   if [ $? == $ERROR_ON_ASSIGN -o $EXIT_STATUS == $OK ] ; then
24     END_EXECUTION= 0 # Imagen ya procesada, continuar con la siguiente.
25   else
26     END_EXECUTION= 1 # Error en la ejecución, finalizar pilot job.
```

Figura 6. Script bash del pilot job que ejecuta en el grid GISELA

```
1  # Obtener una imagen no procesada.
2  query_result=$(mdcli -c mdclient.config "SELECT FILE FROM $path/images WHERE status = '$AVAILABLE' LIMIT 1")
3  if [ "$query_result" != "" ] ; then
4   # Marcar imagen como asignada
5   mdcli -c mdclient.config "updateattr ${path}/images/${query_result} status $ASSIGNED last_update $(date +\ %s)
        last_update_job \"$JOB_ID\" 'status=$AVAILABLE'"
6   if [ $? != 0 ] ; then
7     # Otro job tomó la imagen entre el select y el assign (es raro que suceda)
8     exit $ERROR_ON_ASSIGN
9   # Hallar la mejor réplica y descargar la imagen, como en el script bash del pilot job
10  find_best_replica()
11  lcg-cp --verbose --checksum --checksum-type md5 $BEST_REPLICA ${image_local_path}
12  # Procesar imagen y subir resultados
13  ./Digi-Clima $image_local_path
14  lcg-cr --verbose --checksum --checksum-type md5 --vo prod.vo.eu-eela.eu -l ${path_LFN}/results/$query_result -d
        $DEFAULT_SE $RESULT
15  # Marcar imagen como procesada
16  mdcli -c mdclient.config "updateattr ${path}/images/${query_result} status $PROCESSED last_update $(date +\ %s)
        last_update_job \"$JOB_ID\" 'status=$ASSIGNED"
17 else # No hay más imágenes para procesar, finalizar
18   exit $END
```

Figura 7. Script bash que ejecuta Digi-Clima.

La imagen a procesar se obtiene utilizando el comando `lcg-cp` (línea 11), y luego se ejecuta la aplicación Digi-Clima sobre la imagen obtenida (línea 13). Por último, se almacenan los resultados en el SE utilizando el comando `lcg-cr` (línea 14), y se actualizan los metadatos en AMGA para indicar que la imagen ya fue procesada (línea 16).

3.4. Trabajos independientes en OurGrid

En la actualidad, la implementación de Digi-Clima para ejecutar en OurGrid se encuentra en etapa de desarrollo.

La principal característica de la plataforma OurGrid es su trabajo con workers instalados sobre máquinas virtuales, permitiendo una mayor flexibilidad y escalabilidad en el uso de los recursos computacionales [1].

Existe un prototipo de implementación paralela capaz de procesar un conjunto reducido de imágenes sobre un nodo local de OurGrid instalado en la Facultad de Ingeniería, Universidad de la República, Uruguay. La solución en desarrollo empaqueta el binario de Digi-Clima y las imágenes a ejecutar, y utiliza un JDF cuyo esquema se presenta en la Figura 8 (para una única imagen).

```
1  job :
2  label : job−test
3
4  task :
5  init : put img20110214_13484.jpg img20110214_13484.jpg
6              store digiclima.zip digiclima.zip
7  remote : unzip \$STORAGE/digiclima.zip; ./run_RunAll.
       sh ./MATLAB_Compiler_Runtime/v714
       img20110214_13484.jpg img20110214_13484.jpg
8  final : get img20110214_13484.jpg.out
       img20110214_13484.jpg.out
```

Figura 8. Archivo JDF para ejecutar Digi-Clima en OurGrid (procesa una imagen)

En el JDF del trabajo en OurGrid, las líneas 1-2 identifican al trabajo. Luego, las líneas 4-8 describen una tarea. Ourgrid permite indicar la ejecución de tareas en paralelo, y es posible agregar al JDF una cláusula para indicar requerimientos (por ejemplo, ejecutar en un sistema operativo Linux, o indicar una cantidad mínima de memoria RAM).

La sub-clausula `init` está subordinada a la cláusula `task`, e indica qué archivos serán transferidos desde el *broker* al *worker* antes de ejecutar la tarea. La sub-clausula `remote`, también subordinada a `task`, especifica el comando a ejecutar en el worker, en el caso presntado corresponde a descomprimir el paquete que contiene el binario de Digi-Clima y el runtime de Matlab, y ejecutar Digi-Clima.

Los comandos `get` y `store` utilizados en la cláusula `init` premiten copiar archivos desde el broker al worker. El comando `store` verifica que el archivo no exista en el worker como consecuencia de una copia en una ejecución anterior, mientras que `get` no lo hace.

El JDF presentado ejemplifica lo simple que resulta el uso de la infraestructura, dado que es posible utilizarse el portal de OurGrid enviando el JDF y los archivos `digiclima.zip` y la imagen en cuestión.

4. Evaluación experimental

Esta sección presenta la evaluación experimental de las implementaciones desarrolladas para la ejecución distribuida de Digi-Clima.

4.1. Plataformas de ejecución

La evaluación experimental se realizó utilizando las siguientes plataformas:

- *Cluster FING*: procesadores Intel Xeon E5430/5520 y AMD Opteron 6172, 4/8/12 cores a 2.1/2.25/2.6 GHz, 1 GB de memoria RAM por core, red Gigabit Ethernet (`www.fing.edu.uy/cluster`).
- *GISELA grid*: procesadores Intel Xeon 5400, 4 cores a 2.6 GHz, 2 GB de memoria RAM por core, red Gigabit Ethernet (`www.ciemat.es`).
- *Nodo Ourgrid*: 2 procesadores AMD Opteron 6172, 12 cores a 2.1 GHz, 24 GB memoria RAM, red Gigabit Ethernet (`www.fing.edu.uy/cluster`).

4.2. Maestro-esclavo MPI

La implementación maestro-esclavo en cluster fue la primera desarrollada, y por este motivo su evaluación experimental se realizó con un prototipo reducido de Digi-Clima, que demandaba un tiempo de ejecución inferior a un minuto por imagen, y una base de 1000 imágenes. El tiempo de procesamiento reducido condiciona los resultados, al incrementarse el peso del overhead en la comunicación, así como en la lectura de las imágenes desde el disco. A pesar de estas limitaciones, los resultados obtenidos son valiosos, dado que representan una cota inferior para el desempeño esperado en la ejecución de otras versiones de Digi-Clima.

Se evaluó el tiempo de ejecución del prototipo secuencial de Digi-Clima para procesar 100 imágenes, y de la solución paralela implementada para procesar 100 y 1000 imágenes utilizando diferente número de procesadores (#p) Los resultados se presentan en la Tabla 1.

La Figura 9 presenta un análisis gráfico del speedup algorítmico de la implementación maestro-esclavo MPI en el Cluster FING (puntos rojos) para 1000 imágenes. La línea punteada azul corresponde al speedup algorítmico lineal.

Los resultados indican que el speedup algorítmico es casi lineal al procesar 1000 imágenes, y que importantes desviaciones se detectan al procesar 100 imágenes con más de 20 procesadores. Este resultado refleja la importancia de la relación entre la cantidad de imágenes y el número de procesos esclavos. Los mejores resultados de eficiencia computacional se alcanzan al procesar un gran número de imágenes, ya que en caso contrario los tiempos de inicialización y overhead por comunicaciones impactan en el desempeño.

Tabla 1. Análisis de desempeño: maestro-esclavo MPI en el Cluster FING

#p	100 imágenes		1000 imágenes	
	tiempo	*speedup*	*tiempo*	*speedup*
1	48m 06s	-	-	-
3	15m 10s	3.2	151m 45s	3.2
13	3m 46s	12.7	35m 35s	13.5
23	2m 24s	19.9	20m 39s	23.3
29	-	-	16m 00s	30.1
53	-	-	8m 55s	53.9
73	-	-	6m 39s	72.3

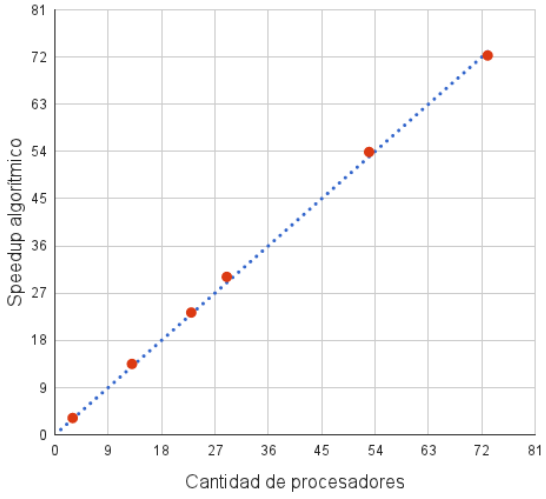

Figura 9. Speedup algorítmico: maestro-esclavo MPI en el Cluster FING

A partir de los resultados expuestos puede concluirse que es conveniente procesar al menos 10 imágenes por esclavo para que la solución alcance un speedup algorítmico lineal. Este valor resulta razonable considerando el objetivo del proyecto de procesar miles de imágenes en una infraestructura de cientos de procesadores.

4.3. Pilot jobs en el Cluster FING

Para estudiar el desempeño de la solución con pilot jobs en el Cluster FING, se realizaron dos evaluaciones:

1. *Experimentos de operativa normal*, enviando trabajos independientes al gestor del Cluster FING hasta alcanzar 100 trabajos concurrentes y analizar el desempeño en dichas condiciones. Este análisis tiene interés desde el punto de vista práctico, dado que refleja las condiciones de una puesta en producción de la solución, permitiendo evaluar su desempeño en las condiciones en que será utilizada.
2. *Experimento en múltiples procesadores de un mismo nodo*, que modifica el descriptor del trabajo ejecutar varios pilot jobs en paralelo en el mismo nodo. Este análisis permite evaluar el balance de cargas en la ejecución y así como tener un valor preciso de tiempo total, y además, permite estresar la memoria caché de tercer nivel de los procesadores del nodo, que es un cuello de botella para la ejecución.

4.3.1. Operación normal con 100 pilot jobs

El objetivo principal del análisis es evaluar el desempeño de la solución en condiciones de carga alta del cluster. Se utilizó una base de datos de 1200 imágenes y se enviaron 100 trabajos al gestor del Cluster FING, que ocuparon un tercio de la infraestructura durante más de dos horas.

La ejecución de Digi-Clima para las 1200 imágenes demandó 2 horas y 10 minutos de cómputo efectivo, sin considerar tiempos de espera en la cola del gestor del cluster. El análisis utilizando un único procesador indicó que el tiempo de ejecución del algoritmo paralelo para una imagen aislada es en promedio un 8 % superior al tiempo del secuencial. Suponiendo que este overhead de paralelismo es representativo dpara todas las imágenes, una proyección indica que el algoritmo ejecutando en un único procesador insume aproximadamente 200 horas. El speedup algorítmico estimado es de de 92.3 al utilizar 100 procesadores.

La Tabla 2 presenta los tiempos de ejecución de las implementaciones consideradas. La ejecución distribuida se realizó sobre un conjunto heterogéneo de procesadores, y la multihilos sobre dos Quad core Xeon E5430 (8 cores).

Tabla 2. Análisis de desempeño: Digi-Clima con pilot jobs en el Cluster FING

algoritmo	#p	imágenes	tiempo	speedup
secuencial	1	1200	200h	1
multi-thread	8	1200	160h	1.25
pilot jobs	100	1200	2h 10m	92.3

Los resultados en la Tabla 2 muestran el pobre desempeño de la implementación multithreading, que sólo es capaz de alcanzar un speedup de 1.25 al trabajar con 8 cores. Como contrapartida, la solución con pilots jobs alcanza valores de eficiencia casi lineal, reportando un valor de spedup de 92.3 al utilizar 100 procesadores.

4.3.2. Experimento en múltiples procesadores de un mismo nodo

Para este estudio se utilizaron 1000 imágenes y 20 pilot jobs en un nodo con dos AMD Opteron 6172 (24 cores).

La Tabla 3 reporta los tiempos de ejecución y el speedup algorítmico obtenidos en este estudio.

Tabla 3. Análisis de desempeño: Digi-Clima en un único nodo del Cluster FING

algoritmo	#p	imágenes	tiempo	speedup
secuencial	1	1000	148h 51m	1
pilot jobs	20	1000	7h 44m	19.25

Los resultados en la Tabla 3 permiten concluir que valores aceptables de eficiencia se obtienen en cada nodo del cluster: el speedup algorítmico es de 19.25 para 20 procesadores (eficiencia computacional del 96.25 %).

4.4. Pilot jobs en el grid de GISELA

El análisis se realizó en recursos del CIEMAT-TIC, España. Se utilizó un banco de 150 imágenes (500 MB) y un prototipo liviano de Digi-Clima. Para simplificar el análisis, se incluyó la restricción de ejecutar en el CIEMAT-TIC y las imágenes se almacenaron en el SE de dicho centro. Esto no es una limitación de la solución implementada, ya que los scripts diseñados soportan la replicación y la búsqueda de imágenes en diferentes SE. De esta manera, los resultados reportados son representativos de ejecuciones en centros de recursos con réplicas locales del banco de imágenes.

Los experimentos utilizaron 4 y 20 pilot jobs para ejecutar Digi-Clima. Los pilot jobs esperaron en promedio 9 minutos para ejecutar, desde que fueron enviados al WMS. Además, a pesar del acceso local a los archivos, entre la descarga de una imagen y el envío de los resultados correspondientes, transcurrieron unos 11 segundos en promedio.

La Tabla 4 presenta los resultados de la evaluación de la versión grid implementada, incluyendo una estimación de los tiempos requeridos al utilizar la versión completa de Digi-Clima (est.) y el speedup correspondiente.

Tabla 4. Análisis de desempeño: Digi-Clima grid en GISELA

escenario	imágenes	# jobs	tiempo	est.	speedup
secuencial	150	1	73.0m	20.0h	1.00
$grid_4$	150	4	32.0m	5.3h	3.77
$grid_{20}$	150	20	12.0m	1.3h	15.05

Los resultados en la Tabla 4 muestran un promisorio comportamiento de speedup casi lineal.

Con la mejora de desempeño obtenida, es posible procesar el banco de imágenes actual del proyecto (30000 imágenes) en 202 horas utilizando 20 pilot jobs, en lugar de las 4000 horas que demandaría procesarlas secuencialmente.

5. Conclusiones y trabajo futuro

Este trabajo ha presentado los avances en el proyecto Digi-Clima Grid. Se han descrito las soluciones paralelas diseñadas e implementadas en el marco del proyecto para ejecutar Digi-Clima sobre infraestructuras cluster y grid. La evaluación experimental analizó los enfoques de paralelismo en cluster y grid, y permitió comprobar que promisorios valores de speedup casi lineal se obtienen al trabajar con Digi-Clima sobre plataformas de memoria distribuida. Con las implementaciones propuestas es posible procesar el total del banco de imágenes del proyecto en un orden de tiempo de días, cuando el procesamiento secuencial demandaría seis meses. Una versión web de la aplicación fue diseñada e instalada en el Scientific Gateway de GISELA [4].

Las principales líneas de trabajo futuro se enfocan en mejorar las técnicas de procesamiento de imágenes, posiblemente incorporando paralelismo con GPUs y estrategias de verificación aplicando human computing. Asimismo, se continuará con el desarrollo de soluciones para la ejecución de Digi-Clima en entornos de grid y cloud computing abordando los temas de mejora de desempeño y tolerancia a fallos, y la utilización del grid/cloud colaborativo OurGrid, para el que existe una implementación en curso.

Referencias

[1] W. Cirne, F. Brasileiro., N. Andrade, L. Costa, A. Andrade, R. Novaes, and M. Mowbray. Labs of the world, unite!!! *Journal of Grid Computing*, 4:225–246, 2006.

[2] Cluster FING. www.fing.edu.uy/cluster, mayo 2013.

[3] Digi-Clima. www.fing.edu.uy/inco/grupos/cecal/hpc/digiclima, mayo 2013.

[4] S. García, S. Iturriaga, S. Nesmachnow, M. da Silva, M. Galnarés, and G. Rodriguez. Developing parallel applications in the GISELA grid infrastructure. In *Proceedings of the Joint GISELA-CHAIN Conference*, pages 9–16, 2012.

[5] R. García-Herrera, G. Können, D. Wheeler, M. Prieto, P. Jones, and F. Koek. Cliwoc: A climatological database for the World's oceans 1750–1854. *Climatic Change*, 73:1–12, 2005.

[6] Grid Infrastructures for e-Science virtual communities in Europe and Latin-America. www.gisela-grid.eu, mayo 2013.

[7] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with Message-Passing Interface*. MIT Press, 1999.

[8] B. Koblitz, N. Santos, and V. Pose. The AMGA metadata service. *Journal of Grid Computing*, 6:61–76, 2008.

[9] Old weather project. oldweather.org, mayo 2013.

[10] C. Wilkinson, S. Woodruff, P. Brohan, S. Claesson, E. Freeman, S. Lubker, C. Marzin, and D. Wheeler. Recovery of logbooks and international marine data: the RECLAIM project. *International Journal of Climatology*, 31(7):968–979, 2011.