UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO

FACULTAD DE CIENCIAS

IMPLEMENTACIÓN DE UN MODELO DE ESTIMACIÓN DE PRECIPITACIÓN PLUVIAL

REPORTE DE SERVICIO SOCIAL

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A:

CARLOS ABRAHAM LAGUNA RUEDA

TUTOR:

ANTONIO CAPELLA KORT

Ciudad Universitaria 2012

Hoja de datos del jurado

1. Datos del alumno

Laguna

Rueda

Carlos Abraham

53 42 27 49

Universidad Nacional Autónoma de México

Facultad de Ciencias

Ciencias de la Computación

303088682

2. Datos del tutor

Dr

Antonio

Capella

Kort

3. Datos del Sinodal 1

Dr

Elisa

Viso

Gurovich

4. Datos del Sinodal 2

Dr

Sergio

Rajsbaum

Gorodesky

5. Datos del Sinodal 3

Lic

Francisco

Solsona

Cruz

6. Datos del Sinodal 4

Mat

José Luis

Torres

Rodríguez

7. Datos del trabajo escrito

Implementación de un modelo de estimación de precipiación pluvial

31 páginas

2012

Agradecimientos

Agradezco a la Facultad de Ciencias y a la Universidad Nacional Autónoma de México por haberme brindado la oportunidad de realizar mis estudios universitarios. Agradezco al Instituto de Matemáticas por haberme otorgado una generosa beca para realizar mi servicio social y trabajo de titulación.

A mi asesor, el Dr. Antonio Capella Kort, por su apoyo en todo momento en la realización de este proyecto que enmarca el último escalón de mi vida universitaria.

A a la Dra. Elisa Viso y al Lic. Francisco Solsona haber aceptado ser sinodales de mi examen profesional y más aún, agradezco todo lo que aprendí de ellos. Sus excelentes cursos fueron para mí la base de mi formación como profesional.

Agradezco a mis padres José Carlos Laguna Rodriguez y Virginia Rueda Pérez, que siempre me han brindado su apoyo incondicional durante mi formación académica.

Agradezco a Adriana porque gracias a su amor, cariño y guía pude expandir mis horizontes, aprender que cada día es un reto nuevo y una oportunidad de ser mejores y, sobre todo, por enseñarme que el cielo es el límite.

Carlos, 2012.

Índice general

Capítulo 1

Introducción

1.1. Antecedentes

En los últimos 50 años la posibilidad de observar la tierra desde el espacio nos ha brindado la oportunidad de cambiar nuestra percepción sobre los sistemas meteorológicos. Gracias al desarrollo de satélites geoestacionarios, la capacidad de observar estos fenómenos y su evolución inició una nueva era en el trabajo científico. En un principio, el simple hecho de poder observar los fenómenos fue un gran paso; con el tiempo, la comunidad científica empezó a trabajar en extraer información cuantitativa de estas nuevas herramientas.

A principios de la década de los 60's los datos meteorológicos, hidrológicos y oceanográficos provenientes de satélites empezaron a tener un mayor impacto en el análisis del medio ambiente. El 7 de Diciembre de 1966, la *National Aereonautics and Space Administration* (NASA) puso en órbita el primer satélite geoestacionario *Applications Technology Satellite* (ATS-1), que tenía la capacidad de observar sistemas meteorológicos en proceso. El satélite ATS-1 era capaz de tomar una imagen completa de la tierra cada media hora. Poco tiempo después, la *National Oceanic and Atmospheric Administration* (NOAA) inició la operación de la serie GOES con el lanzamiento de GOES-1, llevando la investigación y monitorero del clima al siguiente nivel.

1.2. Objetivo General

El objetivo general de este trabajo es la estimación de precipitación pluvial en el sur de México, empleando técnicas que se basan en datos satelitales geoestacionarios y en correcciones a las mismas por medio de lecturas pluviales en tierra.

1.3. Objetivos específicos

- Obtener la información satelital y de las estaciones pluviométricas en tierra para espacios temporales en donde es conocido que ocurrieron eventos importantes de inundaciones.
- Diseñar una base de datos geográfica que permita acceder a la información de manera eficiente.
- Desarrollar un programa en C++/Fortran que se conecte a la base de datos y procese la información para obtener un mapa de lluvias, así como un análisis de los datos obtenidos para saber qué tan confiable es el algoritmo.
- Desarrollar una página web que permita a los investigadores acceder y ejecutar el programa de manera sencilla y amigable.

Capítulo 2

Desarrollo

2.1. Obtener datos satelitales y lecturas pluviométricas

La *National Oceanic and Atmospheric Administration* (NOAA) es una agencia federal Estadounidense que se encarga de monitorear las condiciones oceánicas y atmosféricas por medio de sus satélites. En particular, los satélites geoestacionarios GOES 11 y 12 toman imágenes en infrarrojo de las nubes que están sobre el territorio Mexicano.

Para este proyecto pedimos a la NOAA que nos proporcionara información de una zona amplia del sur del país en intervalos de tiempo, en los cuales se presentaron fenómenos de importancia de inundaciones en la zona de sur de México.

Es necesario que se procese la información que se descarga directamente de los servidores de la NOAA. Para ello se usa una aplicación llamada *NOAAA Weather and Climate Toolkit* para generar archivos de texto plano que contiene toda la información para cada punto leído.

La *Comisión Federal de Electricidad* (CFE) posee una serie de estaciones de medición pluviométricas que registran la lluvia que cae sobre la estación en cada hora. El portal de la CFE permite acceder a las estaciones deseadas y a las fechas disponibles. Desafortunadamente, la información que provee la CFE es limitada para algunas fechas.

2.2. Diseño de la apliación en C++/Fortran

Las aplicaciones que realizan cálculos numéricos complejos requieren que la velocidad de respuesta sea la mínima posible y que la programación de las mismas sea sencilla.

GOES11_BAND_04_20070907_060013.wkt.txt

```
Feature[ id=0 , geom=POINT (-98.96861023891975 12.714290538946788) , value=270.0 ]
Feature[ id=1 , geom=POINT (-98.9390185259909 12.714953698706756) , value=270.0 ]
Feature[ id=2 , geom=POINT (-98.90940946293838 12.715617716209717) , value=270.0 ]
Feature[ id=3 , geom=POINT (-98.87978301101694 12.716282592969165) , value=269.0 ]
Feature[ id=4 , geom=POINT (-98.8501391313556 12.71694833050361) , value=270.0 ]
Feature[ id=5 , geom=POINT (-98.82047778495668 12.717614930336664) , value=270.0 ]
Feature[ id=6 , geom=POINT (-98.79079893269555 12.718282393996851) , value=268.0 ]
Feature[ id=7 , geom=POINT (-98.76110253531925 12.718950723018043) , value=265.0 ]
Feature[ id=8 , geom=POINT (-98.73138855344803 12.719619918938783) , value=262.0 ]
Feature[ id=9 , geom=POINT (-98.701656947571 12.720289983303346) , value=260.0 ]
Feature[ id=10 , geom=POINT (-98.67190767805013 12.72096091766047) , value=260.0 ]
Feature[ id=11 , geom=POINT (-98.64214070511643 12.721632723564387) , value=262.0 ]
Feature[ id=12 , geom=POINT (-98.6123559888695 12.722305402574689) , value=266.0 ]
Feature[ id=13 , geom=POINT (-98.58255348927807 12.722978956256009) , value=268.0 ]
```

Figura 2.1: Archivo de información satelital en texto plano

Históricamente, Fortran ha servido al mundo científico para modelar sistemas complejos. Su sintaxis, muy similar al lenguaje matemático, lo hace sumamente atractivo para desarrollar aplicaciones numéricas. Sin embargo, es su alta especialización lo que permite al compilador hacer optimizaciones eficientes en el código, produciendo programas muy rápidos.

Por otra parte, C/C++ es un lenguaje de propósito general que permite al programador trabajar en tareas que requieren el uso de estructuras de datos complejas y el acceso a los servicios del sistema operativo, por ejemplo: manejar archivos, memoria y hacer conexiones a bases de datos.

La combinación de ambos lenguajes se realiza mediante la definición de métodos externos de C. De esta manera, es posible usar lo mejor de ambos lenguajes obteniendo resultados eficientes usando una programación sencilla.

Entorno de desarrollo

Los entornos de desarrollo integrados (EDI) son herramientas sumamente útiles en la implementación de proyectos de software. Los EDI son aplicaciones que consisten de un editor de código, un compilador, un depurador y una interfaz gráfica.

Para este proyecto se adoptó Netbeans[1] en su versión 7 pues es uno de los EDI más aceptados y fáciles de usar. La integración que posee con C++ y Fortran hace que la programación sea fácil.

[1] www.netbeans.org/

Control de versiones

En cualquier proyecto de desarrollo de software es indispensable contar con un control de versiones del código. Entre las opciones existentes se optó por usar GitHub,[2] pues provee de una plataforma web robusta y sobre todo segura. Para este proyecto se usó un repositorio privado que garantiza que sólo los colaboradores autorizados puedan ver y modificar el código.

Finalmente, existen dos bibliotecas que son utilizadas para realizar consultas a bases de datos MySQL[3]. Una de ellas está implementada en C++ y la otra en C. Se eligió la biblioteca *mysqlclient* implementada en C por su facilidad de uso.

Documentación detallada del programa

La documentación detallada del código se realizó mediante Doxygen[4]. Esta herramienta sirve para realizar documentación de programas escritos en diferentes lenguajes de programación. Al igual que en Javadoc, la documentación se escribe dentro del mismo código como comentarios siendo de esta manera relativamente sencillo de mantenerlo actualizado. Doxygen permite realizar referencias cruzadas entre la documentación y el código, por lo que es muy sencillo ver directamente el código desde la documentación.

El producto final es una página web, un manual que puede ser visto con el comando man de UNIX y un documento PDF que detallan cada archivo, función y módulo usado en el sistema.

2.2.1. Estandarización de la información

Uno de los primeros problemas al manejar la información del satélite fue la naturaleza misma de la información.

Los satélites GOES 11 y 12 entregan imágenes con resoluciones diferentes y, por lo tanto, puntos diferentes para la misma hora; más aún, las imágenes no presentaban necesariamente una forma regular.

El primer paso para manejar de manera confiable la información fue realizar un pre-proceso de la misma de tal manera que se contara con imágenes consistentes y regulares.

La solución adoptada fue dividir el territorio nacional en intervalos predefinidos para generar una malla regular a la cual se podía ajustar la información de los satélites.

[2] https://github.com
[3] www.mysql.com
[4] http://www.doxygen.org/

Figura 2.2: Página web generada por Doxygen

Una vez definida la malla, se subió su información a la base de datos para poder obtener de manera dinámica los puntos que se deben considerar para cualquier polígono.

El siguiente paso fue implementar un algoritmo de interpolación que tomara una imagen satelital arbitraria y ajustara su información a una malla contenedora[5].

Como se ilustra en la Figura 2.3, la información del satélite puede ser muy irregular. Para resolver el problema envolvemos la imagen en la malla e interpolamos la información. Las zonas para las cuales no hay datos disponibles son rellenadas con valores -1, pues se acordó que un valor negativo para las lluvias sería un buen indicador de que no hay información.

El algoritmo de interpolación se describe a continuación:

1. Leer información del satélite y construir un arreglo de las filas que conforman la imagen.

2. Para cada punto x en la malla artificial se obtienen los cuatro puntos de la imagen satelital que forman el cuadilátero más pequeño tal que x está dentro del cuadrilátero.

3. Para realizar la interpolación se requieren sólo tres puntos, por lo que debemos eliminar uno de los puntos del cuadrilátero que contiene al punto x. Un cuadrilátero induce 2 triángulos, por lo que se debe saber dentro de qué triángulo se

[5]Algoritmo implementado por Adrián Tovar

encuentra el punto x. Para saberlo se usa un algoritmo que determina si un punto se encuentra dentro de un polígono usando el Teorema de la curva de Jordan.

4. Finalmente, teniendo el triángulo más pequeño que envuelve a un punto x se puede calcular la ecuación de un plano que pasa por esos tres puntos.

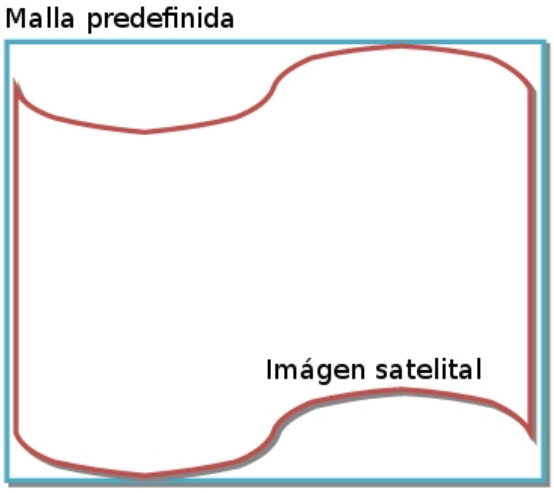


Figura 2.3: Interpolación de las imágenes satelitales

El programa de interpolación fue implementado en Fortran mientras que las subrutinas para pedir la malla adecuada a la base de datos fueron implementadas en C.

2.2.2. Descripción del argoritmo STAR

El algoritmo STAR[1] se encarga de transformar las temperaturas registradas por los satélites y las convierte en una primera estimación pluvial. Recibe como entrada una matriz de temperaturas y para cada punto p de la matriz obtiene una vecindad de radio 50 alrededor del punto. Para esta vecindad calcula la media x' y la desviación estandard σ.

Usando esta información se calculan dos índices llamados RRn y RRc que finalmente son utilizados por el algoritmo STAR

Es importante mencionar que éste proceso se ejecuta para cada lectura del satélite. Es posible que se tengan 3 o 4 lecturas por hora. El algoritmo *Asimilación* requiere que se tenga una sola lectura por hora. Es entonces que se debio realizar un promedio de los archivos si se tienen dos lecturas y ejecutar un algoritmo *trimean* cuando se tengan 3

Algorithm 1 Cálculo del índice RRc

if $temperatura \geq 201$ **then**
  $rrc \leftarrow 1{,}1183 * (100000000000{,}0)^(- 0{,}036382 * temperatura^{1,2})$
else
  $rrc \leftarrow 72$ {máximo valor de lluvia posible}
end if

Algorithm 2 Cálculo del índice RRn

if $temperatura \geq 250$ **then**
  $rrn \leftarrow 0$
else
  $rrn \leftarrow min(\frac{250 - temperatura}{5}, 12)$
end if

Algorithm 3 Algoritmo STAR

for cada punto en la imagen **do**
  $vecindad \leftarrow VecindadRadio50(x, y)$
  $x' \leftarrow Promedio(vecindad)$
  $sigma \leftarrow Desviacion(vecindad)$
  $rrc \leftarrow ComputarRRc((x, y).temperatura)$
  $rrn \leftarrow ComputarRRn((x, y).temperatura)$
  $z = min(\frac{x' - (x,y).temperatura}{sigma}, 1{,}5)$
  if $z > 0$ **then**
    $rr = \frac{rrc * z^2 + rrn * (1{,}5 - z)^2}{z^2 + (1{,}5 - z)^2}$
  else
    $rr = 0$
  end if
end for

lecturas. De esta manera se conserva el mejor valor posible para cada hora. El algoritmo *trimean* es el siguiente:

$$TM = \frac{Q_1 + 2Q_2 + Q_3}{4} \tag{2.1}$$

2.2.3. Descripción e implementación del algoritmo *Asimilación*

El algoritmo *Asimilación* se encarga de realizar un ajuste de los datos calculados usando el algoritmo STAR. A continuación se describe el algoritmo.

1. Descargar la malla predefinida para el polígono solicitado por el usuario.
2. Ejecutar el algoritmo STAR para obtener **hasta** 3 horas previas de la que se desea calcular.
3. Completar la información del paso anterior con -1 en donde no se encuentre información usando la malla predefinida del paso 1.
4. Descargar las lecturas de las estaciones que se ubican dentro del polígoño.
5. Correr el algoritmo *Asimilación-Fortran*[6].
6. En caso de éxito, subir la información nueva a la base de datos y graficarla
7. En caso de fracaso (información insuficiente), se debe graficar sólo la información del algoritmo STAR.
8. Correr el análisis de error puntual para cada estación dentro del polígono.

Algoritmo *Asimilación-Fortran*

El algoritmo *Asimilación-Fortran* realiza los siguientes pasos:

1. Dada la información de las estaciones en tierra, es necesario interpolar el valor obtenido del algoritmo STAR para el mismo punto y así obtener el error: $Eror = ValorReal - ValorSTAR$
2. Se promedian los errores por unidad de lluvia. Los valores posibles de lluvia toman valores en el intervalo de enteros: $[1, 75]$. Los errores son clasificados y promediados usando este intervalo para generar una gráfica entre lluvia y errores.

[6] Algortimo implementado por Adrián Tovar

3. Usando esta gráfica se ajusta una línea con mínimos cuadrados para obtener una pendiente m y una ordenada al origen b, el algoritmo establece que $m \in (-\infty, 2]$ y $b \in [-0{,}1, 0{,}5]$.
   El valor resultado de la asimilación se calcula haciendo uso de las siguientes ecuaciones:
   Se sabe que:
   $ValorAsimilacion - ValorSTAR = Error$
   Por otra parte:
   $Error = ValorStar * m + b$
   por lo tanto
   $ValorAsimilado - ValorSTAR = ValorSTAR * m + b$
   Finalmente
   $ValorAsimilado = (m + 1)ValorSTAR + b$
   Cuando no se puede ajustar una línea porque sólo existe un punto entonces sólo se suma el valor de la coordenada x del punto:
   $ValorAsimilado = ValorSTAR + x(punto)$

El principal problema con este algoritmo es que no es muy funcional si no existe información de las estaciones en tierra o si las lecturas de las estaciones entran dentro del mismo rango de lluvia debido a que el paso 2 promedia las lecturas por unidad de lluvia.

2.3. Diseño de la base de datos

El diseño de la base de datos geográfica responde a una serie de requerimientos operativos:

- Debe guardar cada punto en su contexto espacial y temporal.
- Debe guardar la información de las estaciones de la CFE en su contexto espacial y temporal
- Debe guardar la información de precipitación en cada punto.

El diseño de la base de datos sigue un esquema estrella[2]. El esquema estrella consta de una tabla central, también llamada tabla de hechos, y una serie de tablas, también llamadas tablas de dimensiones, que están relacionadas con la tabla de hechos y la rodean formando una estrella.

El esquema estrella es muy usado en la minería de datos pues es muy simple de entender y es muy veloz para acceder a los datos guardados. Las consultas no son

complicadas desde el punto de vista del usuario final, ya que las condiciones y uniones (JOIN) necesarias sólo involucran a la tabla de hechos y a las de dimensiones.

A pesar de que la minería de datos usa el esquema estrella para realizar análisis multidimiensional, las características mencionadas hacen que este diseño sea el ideal para realizar consultas a la base de datos geográfica.

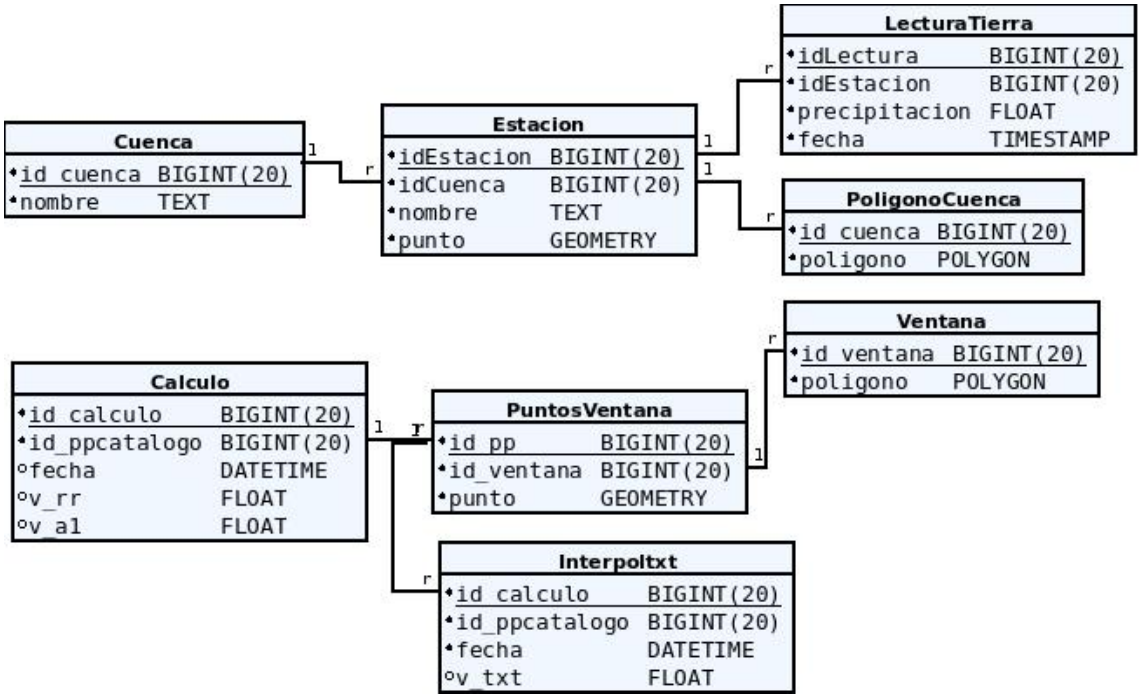


Figura 2.4: Diseño de la Base de Datos Geográfica siguiendo un esquema estrella

El incremento en la velocidad de acceso a la información se logra mediante la aplicación de índices en los campos en donde se hacen las búsquedas más frecuentes.

En este proyecto se relizaron índices sobre diferentes tablas:

- Índice sobre el campo *fecha* en la tabla *LecturaTierra*
- Índice sobre el campo *fecha* en la tabla *Interpoltxt*
- Índice sobre el campo *fecha* en la tabla *Calculo*
- Índice geográfico sobre el campo *punto* en la tabla *Estacion*
- Índice geográfico sobre el campo *punto* en la tabla *PuntosVentana*

2.4. Diseño de la página web

En la actualidad las arquitecturas de software más aceptadas y adoptadas en el desarrollo web implementan la arquitectura llamada Modelo-Vista-Controlador (MVC).

La arquitectura MVC[3] intenta aislar tres componentes importantes en el desarrollo de una aplicación web:

- **Modelo:** Administra el comportamiento y los datos de la aplicación mediante peticiones (usualmente de la vista).
- **Vista:** Permite acceder e interaccionar con el modelo mediante una interfaz fácil de usar.
- **Controlador:** Recibe entradas del usuario e inicia la respuesta correspondiente haciendo llamadas al controlador.

Uno de los marcos de trabajo más ampliamente usados que implementan la arquitectura MVC es Java Server Faces[7] (JSF). En JSF las peticiones del usuario son procesadas por un servidor que carga la vista apropiada, crea el árbol de componentes de la vista y procesa sus eventos. El estado de los componentes de la vista son guardados al final de una petición síncrona o asíncrona al servidor.

El usuario final experimenta una experiencia visual más eriquecedora mediante las llamadas asíncronas al servidor que atienden respuestas en tiempo real en el momento en que se interactúa con la aplicación. El valor agregado de JSF es la existencia de otros marcos de trabajo que incrementan la calidad visual de los componentes de la vista, tal es el caso de PrimeFaces.

La página web del presente proyecto se implementó usando JSF y PrimeFaces[8] para lograr una experiencia visual óptima de tal forma que la aplicación sea fácil de usar y atractiva visualmente.

La página web permite seleccionar el rango de tiempo en donde se va a ejecutar el algoritmo. Se usó Google Maps para ilustrar la distribución geográfica de las estaciones mediante marcas distintivas y la selección de un polígono definido por el usuario. La página principal permite al usuario incluir las estaciones de cuencas disponibles para forzar que su información sea considerada en la correción numérica.

[7] http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html
[8] http://www.primefaces.org/

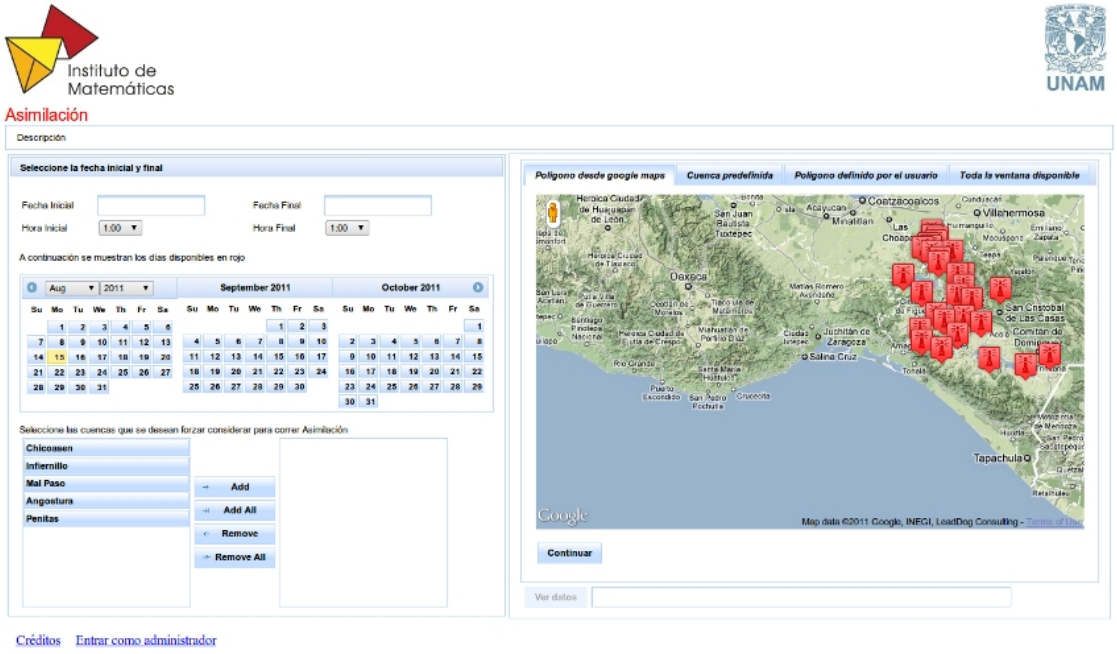


Figura 2.5: http://132.248.17.180:8080/AsimilacionLluvias/

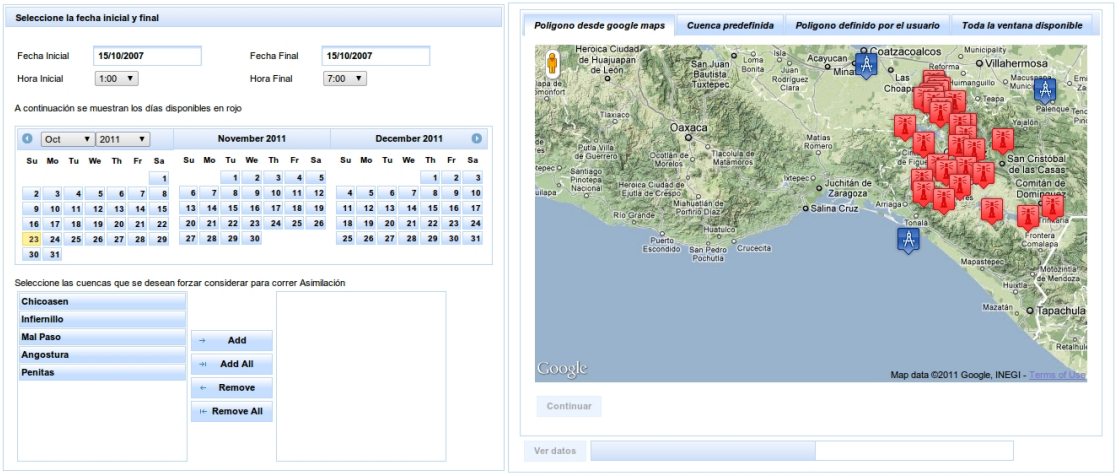


Figura 2.6: Página web ejecutando un cálculo para un periodo de tiempo

2.5. Productos generados

2.5.1. Archivo de lluvia

El programa genera un archivo con terminación .debug que contiene la información calculada de las lluvias. Este archivo puede ser leído por programas de graficación como *gnuplot* y *matlab* para su anlisis.

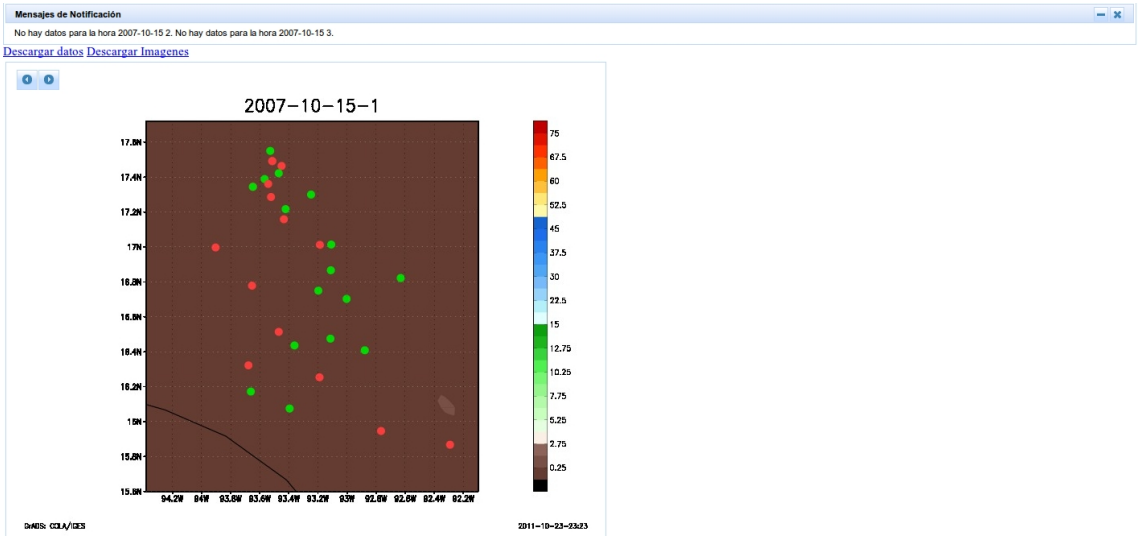


Figura 2.7: Página mostrando el resultado del cálculo solicitado

En la Figura 2.8 podemos observar un ejemplo de un archivo .debug. Este archivo consta de una cabecera que define las esquinas del polígono y una lista de vectores (x, y, precipitación). La lista de vectores define una matriz regular ordenada por renglones.

2007-10-17_01.a1.debug

-98.971832000000006	13.039999999999999	
-91.309860000000000	13.039999999999999	
-98.971832000000006	18.920000000000002	
-91.309860000000000	18.920000000000002	
-98.971832000000006	13.039999999999999	-1.0000000000000000
-98.943664999999996	13.039999999999999	-1.0000000000000000
-98.915488999999994	13.039999999999999	0.0000000000000000
-98.887321000000000	13.039999999999999	0.0000000000000000
-98.859154000000004	13.039999999999999	0.0000000000000000
-98.830985999999996	13.039999999999999	0.0000000000000000
-98.802818000000002	13.039999999999999	0.0000000000000000
-98.774651000000006	13.039999999999999	0.0000000000000000
-98.746475000000004	13.039999999999999	0.0000000000000000
-98.718306999999996	13.039999999999999	0.0000000000000000
-98.690140000000000	13.039999999999999	0.0000000000000000

Figura 2.8: Archivo de lluvia

2.5.2. Archivo de estimación puntual y pruebas de confiabilidad

El algoritmo que utiliza los valores de las estaciones de la CFE permite generar un mapa de lluvias dentro de un polígono determinado. La selección del polígono determina qué estaciones se deberán usar para calcular la corrección numérica.

Estas premisas dan pie a analizar de manera puntual cómo se han ido mejorando las estimaciones pluviales al aplicar consecutivamente los algoritmos; es decir, mediante interpolaciones numéricas nos fue posible comparar la precipitación real reportada por las estaciones de la CFE, contra el resultado del algoritmo STAR y el resultado de la asimilación.

El archivo de estimación puntual describe el comportamiento de los dos algoritmos de estimación pluvial y los compara contra la precipitación real reportada por las estaciones de la CFE.

El archivo de estimación puntual es un archivo .csv que contiene los siguientes campos por estación: **Nombre de la estación, Longitud, Latitud , Valor real, Valor rain rate, Valor asimilado.**

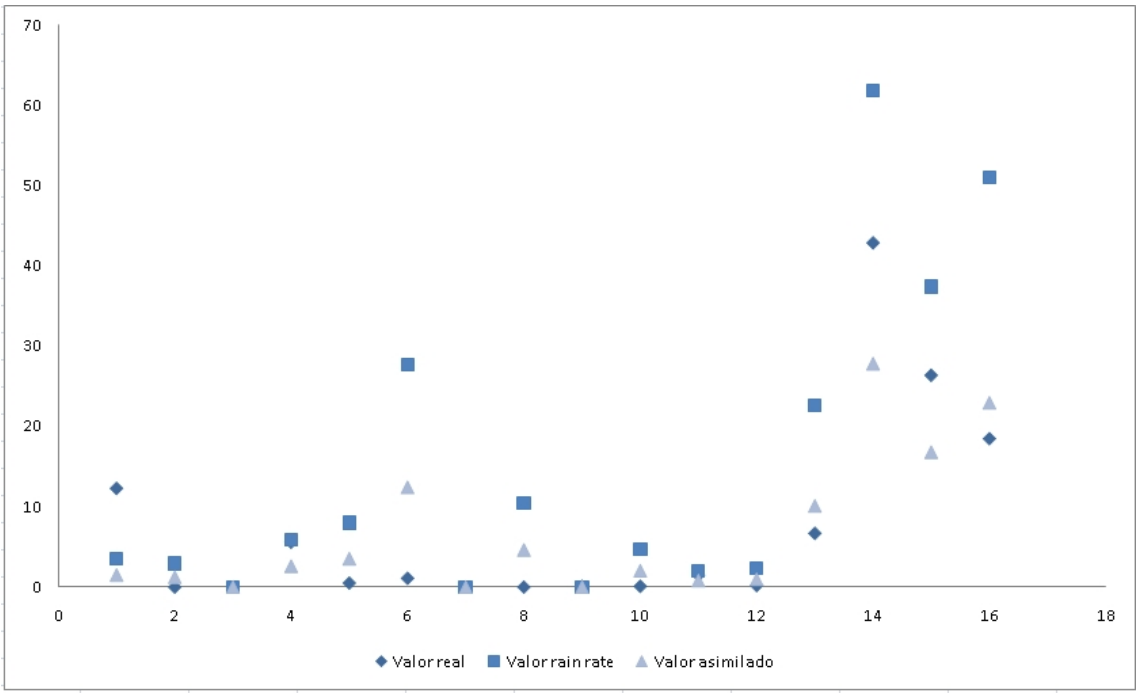


Figura 2.9: Diagrama de dispersión

En la figura 2.9 se muestra un diagrama de dispersión de las predicciones realizadas en cada paso ejecutado por el sistema. En el eje y se representa la precipitación en una hora en particular, en el eje x se representan las estaciones que caen dentro de un polígono determinado.

2.5.3. Mapas de lluvia

Para poder presentar una imagen de la estimación pluvial probamos diferentes programas de graficación, entre los cuales destacan *gnuplot* y GrADS[9].

Se eligió usar GrADS, diseñado originalmente por *NASA Advanced Information Systems Research Program*, pues es un programa de propósito específico para manipular y visualizar datos generados por las ciencias de la tierra. Por otra parte, *gnuplot* es un programa de uso generar para graficar gran variedad de datos científicos, lo cual lo hace poco eficiente para graficar nuestra información, en particular, el uso de curvas de nivel fue privativo en tiempo de ejecución y en nivel de detalle de la imagen.

Se puede apeciar en la Figura 2.7 que GrADS nos permite integrar la ubicación y estado de las estaciones, mostrando en rojo aquéllas que no poseen información, en verde las que sí la poseen y en amarillo las que tienen información incompleta. Adicionalmente, el graficador[10] muestra la silueta de la República Mexicana en negro y una escala de colores lateral que modela la cantidad de lluvia que está cayendo sobre una región específica.

Directamente, la imagen generada por GrADS es un archivo EPS *Encapsulated PostScript*; sin embargo, a pesar de la buena calidad de las imágenes EPS, se debió convertir al formato jpg para conservar la compatibilidad con los navegadores web.

La generación de imágenes fue de vital importancia para poder notar detalles decisivos entre una cantidad enorme de datos. El uso de imágenes nos permitió observar que las imágenes satelitales que provee la *NOAA* abarcaban una zona mucho más pequeña que la que se solicitó. La Figura 2.11 muestra una imagen que no posee suficiente información

[9] http://www.iges.org/grads/

[10] Módulo implementado por Adrán Tovar

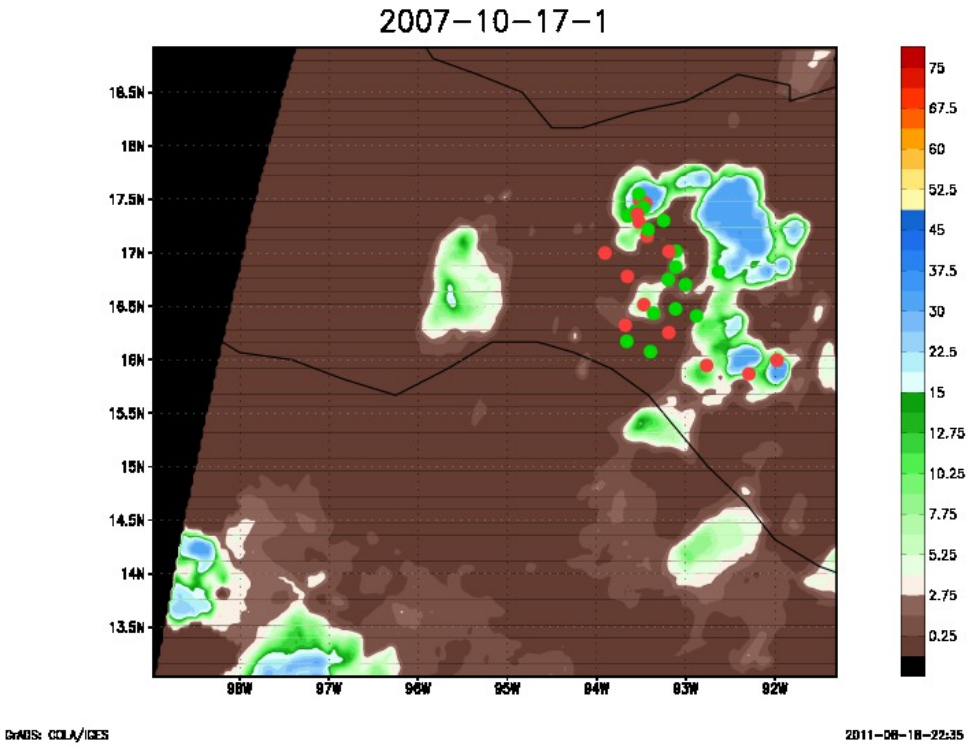


Figura 2.10: Imagen generada por GrADS de la ventana completa de datos

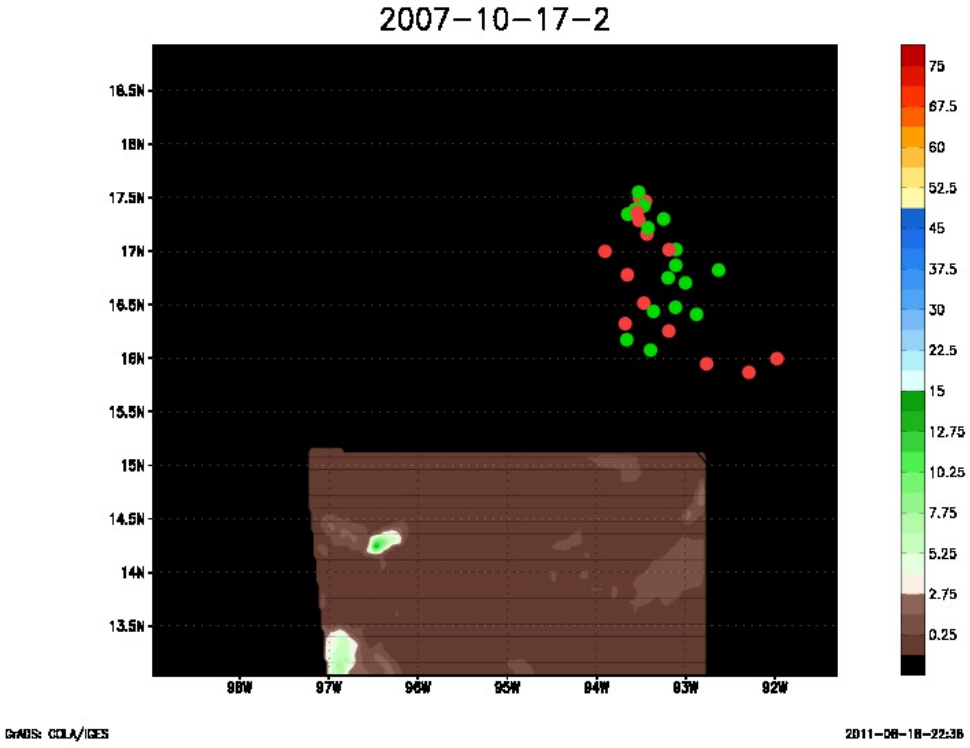


Figura 2.11: Imagen generada por GrADS de la ventana completa de datos con información faltante

Capítulo 3

Manuales de uso e instalación

3.1. Dependencias

Las dependencias del sistema pueden clasificarse en dos tipos:

Compilación

Para compilar el programa es necesario tener instalados los siguientes programas:

- g+
- fortran
- Biblioteca *mysqlconnector* para C

Operación

- Paquete de programas *grads* version 2.0.a7.1 o superior.
- *Apache tomcat 7* [1] o superior o algn otro servidor web que soporte *JSF* [2] y *PrimeFaces* [3].
- *Convert* [4]. Este programa se usa para transformar las imágenes de lluvias en archivos jpeg.

[1] www.apache.org
[2] http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html
[3] primefaces.org
[4] http://www.imagemagick.org/script/convert.php

3.2. Instalación

El programa final se compila usando la herramienta *make*. Se deben tener las bibliotecas instaladas en una ruta disponible para el compilador gcc. El desarrollo de la página web se realizó usando Netbeans 7.0. La instalación de la página en el servidor Tomcat requiere que simplemente se copie el archivo *AsimilacionLluvias.war* en la carpeta de aplicaciones de Tomcat. Posteriormente el servidor desempaquetará el archivo y la apliación podrá ser accedida.

La aplicación web usa internamente el programa en C++/Fortran para realizar los cálculos numéricos. Por lo que hay que copiar la aplicación numérica en la dirección *AsimilacionLluvias/web/Asimilacion* y además se deben poner los permisos adecuados para que el servidor web pueda ejecutar la aplicación.

3.3. Preparación de la Base de Datos

Archivos separados por comas

El primer paso es crear los archivos csv que se subirán a la base de datos. Existen tres tipos de archivos que pueden ser usados:

Archivo de Cuencas

Este archivo es sumamente sencillo pues consta de un identificador numérico único y el nombre de la cuenca.

Archivo de lecuras de estaciones en tierra

Este archivo consta de los siguientes campos: identificador único, identificador de la estación que realizó la lectura, precipitación y fecha en el formato YYYY-MM-DD HH:00:00.

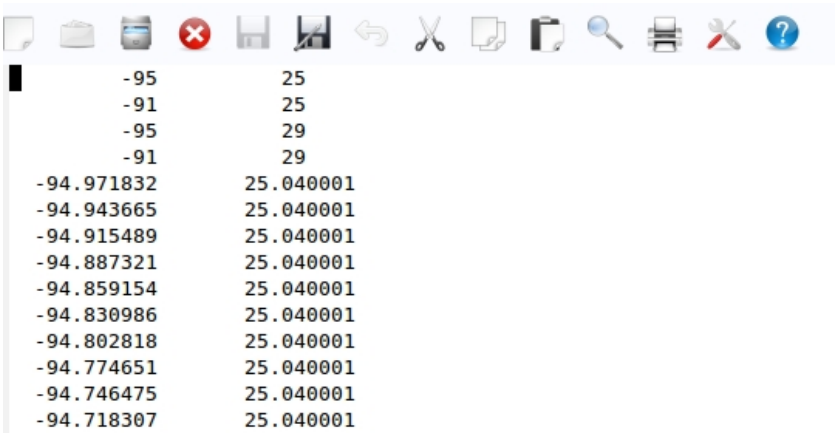
Archivo de definición de estaciones

El archivo que define las estaciones está ligado con las cuencas disponibles. Una estación siempre pertenecerá a una cuenca. Los campos de este archivo son: identificador de cuenca, nombre, longitud, latitud.

Archivos que definen la malla artificial

Para poder realizar las interpolaciones y guardar la información descargada en el satélite se crearon una serie de archivos que definen una malla artificial sobre la República Mexicana. La Figura 3.1 muestra uno de los 45 archivos archivos de texto que define la

malla artificial; estos archivos no deben modificarse.



-95	25
-91	25
-95	29
-91	29
-94.971832	25.040001
-94.943665	25.040001
-94.915489	25.040001
-94.887321	25.040001
-94.859154	25.040001
-94.830986	25.040001
-94.802818	25.040001
-94.774651	25.040001
-94.746475	25.040001
-94.718307	25.040001

Figura 3.1: Malla artificial para un sector de la Repblica Mexicana

Creación de la base de datos y carga de contenido

Es necesario que se posean los privilegios suficientes de administrador para poder realizar el proceso de instalación y preparación que se resume en los siguientes pasos:

1. Crear la base de datos *Asimilacion1* usando MySQL
2. Ejecutar el script que crea el esquema de la base de datos con la instrucción
   `mysql Asimilacion1 <Asimilacion1.1.sql`
3. Cargar la información de las cuencas
   `.\asimilaciondb_v1 --cuenca Cuenca.csv`
4. Cargar la información de las estaciones
   `.\asimilaciondb_v1 --station Estaciones.csv`
5. Cargar la información de la malla artificial ejecutando el script de carga
   `.\loadMallaArtificial.sh`

3.4. Uso de la aplicación

El diseño del programa contempló que la usabilidad de la aplicación fuera igual al de las herramientas de línea de comandos de GNU. De esta manera, la curva de aprendizaje del usuario final sería muy reducida, especialmente para aquellas personas que están familiarizados con este ambiente de desarrollo. Para lograr esto se usó la biblioteca *getopt.h* que es estándard de los sistemas UNIX.

Programa *asimilacion2*

A continuación se lista la serie de opciones de la aplicación

- -txt [ARCHIVO]	Sube a la base de datos un archivo de texto del satélite y verifica que la información no se haya guardado previamente
- -txtu [ARCHIVO]	Sube a la base de datos un archivo de texto del satélite sin verificar que la información sea consistente. Esta opción permite la carga más rápida del archivo, pero se debe estar seguro que la información del mismo no esté previamente en la base de datos
- -a1	Indica que se va a calcular el algoritmo *Asimilación*
- -rr	Indica que se quiere calcular la primera estimación de lluvia
- -fpolygon [ARCHIVO]	Indica dónde está guardado el archivo que define el polígono que se usará para realizar el cálculo
- -date [YYYY-MM-DD HH]	Indica qué fecha y hora se desea calcular
- -outputdir [ARCHIVO]	Indica en qué directorio se van a guardar los productos del cálculo
- -asHour [N]	Indica que se usarán N horas previas para correr *Asimilación*

Programa asimilaciondb_v1

A continuación se lista la serie de opciones de la aplicación.

- -station [ARCHIVO]	Sube a la base de datos un archivo que tiene las definiciones de las estaciones.
- -cuenca [ARCHIVO]	Sube a la base de datos un archivo csv que tiene definiciones de las cuencas
- -window [ARCHIVO]	Sube un archivo que define una sección de la malla predefinida

Capítulo 4

Conclusiones

4.1. Trabajo futuro y escalabilidad

4.1.1. Paralelismo

Introducción

La proliferación de sistemas de alto rendimiento y el surgimiento de redes cada vez más veloces han traído un gran interés en los sistemas parelos y distribuídos.

Desde hace algunos años las tecnologías de cómputo han sufrido un cambio significativo principalmente en la arquitectura. La imposibilidad de incrementar la velocidad del reloj interno de las computadoras debido al sobrecalentamiento y alto consumo energético de los procesadores llevó a la industría a dar por terminada la carrera de los ciclos de reloj. Es así como surge la era de los procesadores multinúcleo.

La nueva estrategia consiste en tener múltiples procesadores integrados en la misma estampa compartiendo los mismos cachés de memoria o segmentando el caché para cada procesador individual.

Las nuevas aplicaciones deben ser diseñadas y programadas de tal manera que exploten los beneficios de las nuevas arquitecturas, es decir, coordinar procesos para distribuir el trabajo, para así lograr mejorar el desempeño de los programas.

La computación en paralelo ha tenido un impacto amplio en una gran variedad de áreas, desde simulaciones computacionales para aplicaciones científicas y de ingeniería hasta aplicaciones comerciales en minería de datos y procesamiento de transacciones.

Clasificación de paralelismo

Las computadoras concurrentes permiten la explotación de muchos modelos de paralelismo. Entre los modelos de paralelismo que se usan comunmente se encuentran los siguientes:

- Procesamiento multihilos.
- Memoria compartida (sin multihilos).
- Memoria distribuida / Paso de mensajes.
- Paralelismo de datos *Single Program Multiple Data* (SPMD)
- Paralelismo de datos e instrucciones *Multiple Program Multiple Data* (MPMD)

Aplicación del paralelismo para un trabajo futuro

Como hemos visto, la implementación actual maneja una zona espacial amplia pero muy determinada en la que el programa puede realizar el cálculo de precipitación.

Un trabajo futuro implica ampliar la zona y el número de horas guardadas en la base de datos. La gran cantidad de información y el requerimiento de un sistema que responda en tiempo real invita a implementar el sistema usando el paradigma SPMD en combinación con programación multihilos.

La naturaleza del algoritmo permite que se pueda operar una imagen individual usando un sistema de memoria compartida por una cantidad de hilos, que permitan calcular de manera independiente secciones de cada imagen.

Escalabilidad de la base de datos

Se realizó un análisis del desempeño del programa para saber qué actividades son las que toman más tiempo en el procesamiento de una imagen satelital. El resultado fue que la activdad más demandante es el acceso a la información de la base de datos lo cual incluye la consulta de puntos dentro de un polígono para un tiempo determinado y la transferencia de los datos para ser procesados.

Las bases de datos comerciales como Oracle tienen una velocidad y eficiencia mucho mejor que MySQL. Estas mejoras se logran por medio de una implementación inteligente que combina no sólo la velocidad del procesador sino también la arquitectura en la cuál se monta la base de datos.

En un trabajo futuro en el que se maneje una cantidad de información mucho mayor, el papel del sistema manejador que se seleccione será de vital importancia.

Bibliografía

[1] http://www.star.nesdis.noaa.gov/smcd/emb/ff/HEtechnique.php

[2] Jiawei Han et. al. *Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems)* Elsevier, 2011.

[3] Ed Burns et. al. *JavaServer faces 2.0: the complete reference* McGraw-Hill, 2009.