ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

DESARROLLO DE UNA APLICACIÓN GRID USANDO GLOBUS TOOLKIT 4

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

VIVIANA MARIBEL CAIZA GARCÍA
JOSÉ VLADIMIR VILLACRESES SÁNCHEZ

DIRECTOR: PhD. ENRIQUE MAFLA

Quito, Febrero 2007

DECLARACIÓN

Nosotros, Viviana Maribel Caiza García y José Vladimir Villacreses Sánchez, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado por ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la siguiente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por su normativa institucional vigente.

Viviana Maribel Caiza García | **José Vladimir Villacreses Sánchez**

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Viviana Maribel Caiza García y José Vladimir Villacreses Sánchez, bajo mi supervisión.

PhD. Enrique Mafla
DIRECTOR DE PROYECTO

DEDICATORIA

A mi esposo, José, el amor de mi vida, quien con su amor, paciencia y dedicación constante logró que cumpla una meta más.
A mi padres, Javier y Gricelda, por incentivarme a seguir adelante y apoyarme en cada momento.
A mis hermanos, Daisy e Isaías, que son mi impulso, por su amor, comprensión y apoyo incondicional.

Mary

A mi esposa y a mis padres.

José

AGRADECIMIENTOS

Agradezco a Dios, por ser el centro de mi vida, por darme fuerzas, inteligencia y sabiduría.
A mi esposo, por impulsarme a seguir, por estar a mi lado y por enseñarme a vencer las dificultades.
A mis padres, por su constante esfuerzo, su sacrifico diario, por toda la confianza que pusieron en mí, y por los valores de responsabilidad que me enseñaron, los cuáles fueron cruciales para alcanzar este logro.
A mis suegros, José y Felisa, por su enorme ayuda, ejemplo y apoyo infinito.
Al Dr. Enrique Mafla, por la acertada tutoría y la confianza brindada para la elaboración de este proyecto de titulación.
Al Ing. Gustavo Samaniego, por su valioso aporte y ayuda incondicional.
A la Ing. Myriam Hernández, por sus consejos y aporte en este proyecto.
A CLIRSEN, por su acertada guía y colaboración que nos permitieron cumplir con los objetivos de este proyecto de titulación.
A mis amigos quienes siempre me animaron a continuar.

Mary

Gracias a todos quienes aportaron en algo a este trabajo. Gracias principalmente a: mi esposa y mi familia, el Dr. Enrique Mafla, la Ing. Janina Olmedo, el capitán Rafael Delgado, el Ing. Gustavo Samaniego, la Ing. Myriam Hernández y a mis amigos.

José

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Grid Computing se enfoca en la compartición coordinada de recursos en "Organizaciones Virtuales" multi-institucionales y dinámicas. Una Organización Virtual está conformada por las instituciones que comparten recursos según políticas de acceso y de seguridad.

Globus Toolkit 4 permite implementar la infraestructura de software de un Grid y el desarrollo de aplicaciones Grid.

El aporte de nuestra tesis será el desarrollo de una aplicación grid y la construcción de la infraestructura grid necesaria básica para demostrar el funcionamiento de la aplicación usando Globus Toolkit 4. Para la selección de la aplicación a desarrollar nos basaremos en un análisis de las facilidades y capacidades de Globus Toolkit 4 para Grids y aplicaciones Grids computacionales. La aplicación ilustra la distribución de procesamiento a diferentes recursos y el ajuste del número de procesos según la demanda de procesamiento, en un ambiente controlado por una sola administración de recursos, tal que todos los recursos tienen aplicadas las mismas políticas de acceso y seguridad.

INTRODUCCIÓN

Esta tesis contempla el análisis de las facilidades y capacidades de “Globus Toolkit 4” para desarrollar aplicaciones grid computacionales, y la aplicación de lo investigado en la selección y desarrollo de una aplicación grid de procesamiento distribuido.

Este documento está dividido en 5 capítulos.

El CAPÍTULO I, “Marco teórico“, contiene teoría fundamental para la tesis, el resultado del análisis de las facilidades y capacidades de Globus Toolkit 4 para desarrollar aplicaciones grid computacionales y la metodología de desarrollo, incluyendo los parámetros usados en la selección de la aplicación a ser desarrollada.

El CAPÍTULO II, “Análisis de requerimientos de la aplicación”, el CAPÍTULO III, “Diseño de la aplicación”, y el CAPÍTULO IV, “Construcción de la aplicación”, abarcan la documentación final del análisis, diseño y construcción de la aplicación seleccionada.

El CAPÍTULO V, “Conclusiones y recomendaciones”, contiene las conclusiones y recomendaciones obtenidas durante la realización de la tesis.

CAPÍTULO I

MARCO TEÓRICO

En esta sección se presenta teoría fundamental para el entendimiento de los capítulos siguientes, el resultado del análisis de las facilidades y capacidades de Globus Toolkit 4 (GT4)[1] para desarrollar aplicaciones grid computacionales y la metodología de desarrollo.

1.1 COMPUTACIÓN DISTRIBUIDA

En este punto se hace la diferenciación entre la computación Grid y otros tipos de computación similares con el fin de que el lector aclare sus ideas respecto a la computación Grid.

Computación distribuida es un modelo de computación en el que el algoritmo está distribuido en varios programas, en distintos espacios de memoria, que se comunican entre sí. Generalmente esto significa que el procesamiento ocurre en nodos diferentes de una red.

Computación paralela es un modelo de computación que consiste de múltiples procesadores en comunicación cercana, normalmente ubicados dentro de la misma máquina. Puede ser de multiprocesamiento simétrico, que tiene múltiples procesadores compartiendo una misma memoria y sistema operativo, o de procesamiento masivamente paralelo, que tiene múltiples procesadores accediendo a su propia memoria y sistema operativo.

La diferencia principal entre la computación paralela de multiprocesamiento simétrico y la computación distribuida está en el acceso a los espacios de memoria.

[1] En adelante GT4 se usará como abreviación de Globus Toolkit 4.

Computación de cluster es un modelo de computación en el que se juntan las capacidades de varios computadores estándar. Un cluster posee software para hacerlo actuar como un sistema paralelo y puede ser visto como un solo sistema.

Las diferencias principales entre computación de clusters y computación Grid son:

- En la computación de clusters la asignación de recursos es ejecutada por un gestionador de recursos centralizado y todos los nodos trabajan juntos como un solo recurso unificado. En la computación Grid cada nodo tiene su propio gestionador de recursos y no se provee una vista de sistema único.
- Los clusters necesitan proximidad física y homogeneidad operativa, un Grid está pensado en que los sistemas que lo conforman son heterogéneos y se encuentran distribuidos geográficamente.
- Los computadores que conforman un cluster poseen una relación de confianza plena, en cambio un grid está formado por computadores que no confían completamente entre sí.

Un cluster puede formar parte de los recursos que integran un grid.

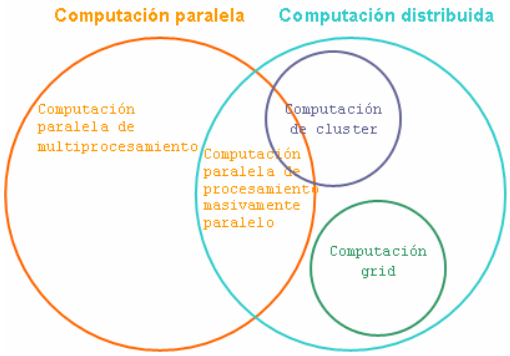

Figura 1.1 Relación entre tipos de computación.

En la Figura 1.1 se muestra la relación entre los tipos de computación mencionados.

1.2 COMPUTACIÓN GRID

En este punto se detallan conceptos e ideas fundamentales dentro de la computación Grid con el fin de que el lector entienda apropiadamente los fundamentos de GT4.

La computación Grid tiene que ver con la compartición coordinada de recursos y la resolución de problemas en Organizaciones Virtuales multi-institucionales y dinámicas [5 Pág. 2]. Lo que le distingue a la computación Grid de otros tipos de computación distribuida es el enfoque en la compartición de recursos tomando en cuenta los requerimientos de las Organizaciones Virtuales. Entre las particularidades de las Organizaciones Virtuales están: los recursos pueden pertenecer a distintas organizaciones o individuos; los recursos son heterogéneos; los recursos pueden estar geográficamente muy alejados entre sí; no existe una relación de confianza plena entre los miembros de la Organización Virtual.

1.2.1 ORGANIZACIÓN VIRTUAL

Una Organización Virtual está conformada por los individuos y/o instituciones que comparten recursos según políticas de acceso y seguridad. Los entes (universidad, empresa, departamento, individuo) involucrados en una Organización Virtual pueden tener distintos grados de relaciones entre ellos.

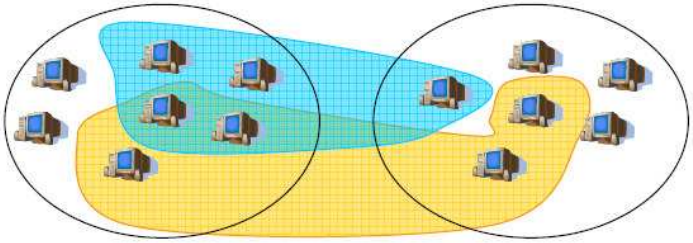

Figura 1.2 Organizaciones Virtuales [15 Pág. 2].

En la Figura 1.2 las áreas irregulares representan Organizaciones Virtuales y los óvalos representan entidades del mundo real.

Una Organización Virtual podría estar formada por dos empresas constructoras involucradas en un proyecto de construcción de un aeropuerto. Sin tener ningún grado de confianza entre ellas comparten sus recursos para el uso de software de simulación y de diseño.

1.2.2 GRIDS

Se encuentran principalmente dos definiciones de un Grid.
Un Grid es un sistema que coordina recursos que no están sujetos a control centralizado, usando protocolos e interfaces de propósito general, abiertos y estándares para proporcionar calidades de servicio no triviales [7 Pág. 2]. Un Grid está enfocado en el uso de recursos en ambientes controlados por distintas administraciones. Los protocolos e interfaces estándares permiten la interoperabilidad. Un Grid debe proporcionar calidades de servicio relativas a, por ejemplo, la disponibilidad, tiempo de respuesta, rendimiento.

Un Grid es una infraestructura de hardware y software que provee acceso seguro, consistente, penetrante y barato a capacidades computacionales sofisticadas [3 Pág. 18]. Un Grid maneja un conjunto de recursos (cómputo, datos, sensores), para esto se requiere infraestructura de hardware para las interconexiones e infraestructura de software para su funcionamiento. La consistencia se refiere a que los servicios y sus interfaces sean estándares para lograr que las aplicaciones desarrolladas sean interoperables. La penetración implica que siempre se cuente con acceso universal a los servicios disponibles dentro de los límites de cualquier ambiente Grid. Un Grid debe ser de acceso barato para que se lo use ampliamente.

1.2.2.1 Tipos de Grids

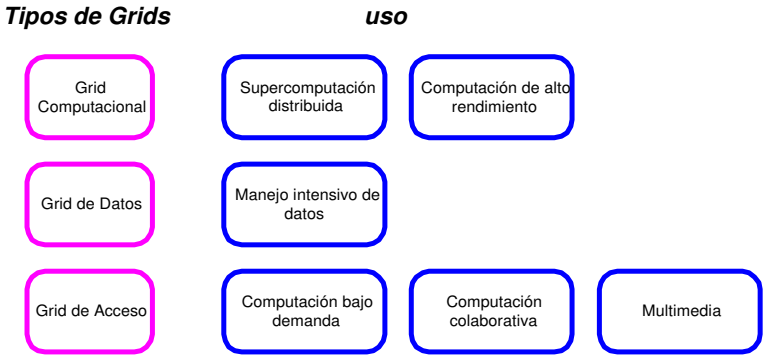

Figura 1.3 Tipos de Grids y sus usos.

Como se muestra en la Figura 1.3, los tipos de Grids y sus usos son:

- Grid computacional: conecta recursos para computación.
  - Supercomputación distribuida: usa Grids para agregar recursos computacionales substanciales para resolver problemas que, por sus demandas, no pueden ser resueltos en un solo sistema.
  - Computación de alto rendimiento: usa Grids para realizar muchas tareas poco acopladas o independientes.
- Grid de datos: integra recursos de datos distribuidos geográficamente.
  - Manejo intensivo de datos: usa Grids para sintetizar nueva información desde datos mantenidos en repositorios, librerías digitales y bases de datos distribuidos geográficamente.
- Grid de acceso: ejecuta tareas que, por concepto, un solo sistema no puede.
  - Computación bajo demanda: usa Grids para satisfacer requerimientos de término corto para recursos que no pueden ser convenientemente ubicados localmente.
  - Computación colaborativa: usa Grids para posibilitar y mejorar interacciones humano-humano.
  - Multimedia: usa Grids como infraestructura para capacidades de tiempo real.

1.2.3 ARQUITECTURA DE SERVICIOS WEB ABIERTA (OGSA)

OGSA define un conjunto de capacidades y comportamientos que se encargan de asuntos clave en un sistema Grid [11 Pág. 4].

OGSA posibilita la interoperabilidad entre recursos distribuidos y heterogéneos [11 Pág. 6]. Los ambientes Grid tienden a ser heterogéneos pasando por una variedad de ambientes de hosting (J2EE, .NET), sistemas operativos (Unix, Linux, Windows, sistemas embebidos), dispositivos (computadores, sensores, instrumentos, sistemas de almacenamiento).

OGSA enfoca el manejo de recursos en términos de servicios, sus interfaces, el estado individual y colectivo de sus recursos, y la interacción de estos dentro de una arquitectura orientada a servicios [11 Pág. 13].

OGSA no enfoca nada relativo a la implementación, requerimientos de funcionamiento, ejecución, ambiente de un servicio, es decir, solo especifica las interfaces, no el comportamiento.

1.2.3.1 Servicios Grid

OGSA representa todo (recursos computacionales, recursos de almacenamiento, redes, programas, bases de datos) como un servicio Grid.
Un servicio Grid es un servicio Web dinámico que sigue convenciones y soporta interfaces estándares para propósitos de manejo de estado [6 Pág. 18].

1.3 GLOBUS TOOLKIT

En este punto se presentan los conceptos fundamentales dentro de GT4 y el resultado del análisis de las facilidades y capacidades de GT4 para desarrollar aplicaciones grid computacionales.

Globus Toolkit es un conjunto de servicios, clientes, herramientas, contenedores y librerías de software de arquitectura abierta, comunitario,

Figura 1.4 Relación entre OGSA y WSRF [15]

En la Figura 1.4 se muestran la relación entre OGSA y WSRF.

El enfoque WS-Resource sirve para modelar estado en un contexto de servicios Web. Un WS-Resource se define como la composición de un servicio Web y su recurso con estado. El WS-Resource framework permite que los WS-Resources sean declarados, creados, accedidos, monitoreados, y destruidos vía mecanismos de servicios Web convencionales. WS-Resource framework es un conjunto de cinco especificaciones técnicas que definen el enfoque WS-Resource.

1.3.1.2 WS-Notification

El patrón de interacción basado en notificación es comúnmente usado en un contexto de servicios Web. WS-Notification estandariza los conceptos e intercambio de mensaje que los esquemas WSDL y XML requieren para expresar el patrón.

1.3.1.3 WS-Addressing

WS-Addresssing provee mecanismos para direccionar servicios Web y mensajes; define endpoint references que comunican la información de la dirección en un formato uniforme que puede ser procesado independientemente del transporte o la aplicación. Esta especificación permite que los sistemas de mensajería soporten la transmisión de mensajes a través de redes con nodos procesantes (firewalls, gateways, proxies).

1.3.1.4 Arquitectura de GT4.

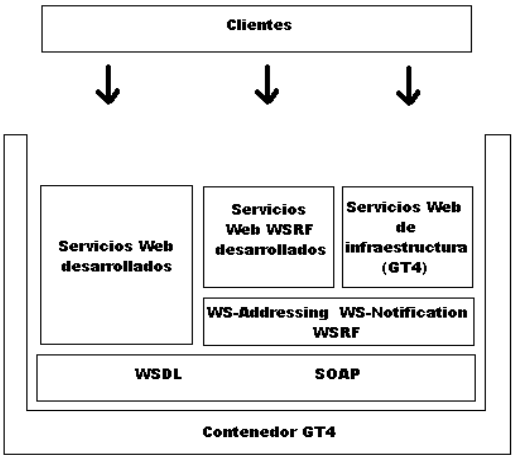

Figura 1.5 Arquitectura de GT4.

Como se muestra en la figura 1.5, la arquitectura de GT4 está esta formada por:

- Servicios de infraestructura Grid. Conforman la infraestructura de software que usan las aplicaciones Grid. Los principales son:
  - En manejo de elementos computacionales y ejecución de actividades sobre esos elementos: Grid Resource Allocation and Management Service (GRAM).
  - En manejo de transferencia de datos: Reliable Transfer Service (RFT).
  - En monitoreo y descubrimiento: Index, Trigger, WebMDS.
- Contenedores de servicios Web. Para deployar servicios, tanto de infraestructura como desarrollados por el usuario, escritos en Java, C y Python.
- Clientes. GT4 provee librerías y herramientas para desarrollar clientes.
- Servicios desarrollados por el usuario. GT4 provee librerías y herramientas para crear servicios.

1.3.2 ANÁLISIS DE LOS COMPONENTES DE GT4[2]

GT4 provee componentes necesarios para construir la infraestructura de software de un Grid computacional (servicios de infraestructura) y aplicaciones que usen esos servicios de infraestructura.

Una vez analizadas las facilidades y capacidades de los componentes de GT4 para el desarrollo de aplicaciones grid computacionales en base a servicios Web, en esta sección se presenta el resultado de este análisis.

Este análisis fue realizado desde el punto de vista del desarrollador de aplicaciones (funcional, uso de servicios y facilidades), no desde el punto de vista del administrador (instalación, configuración).

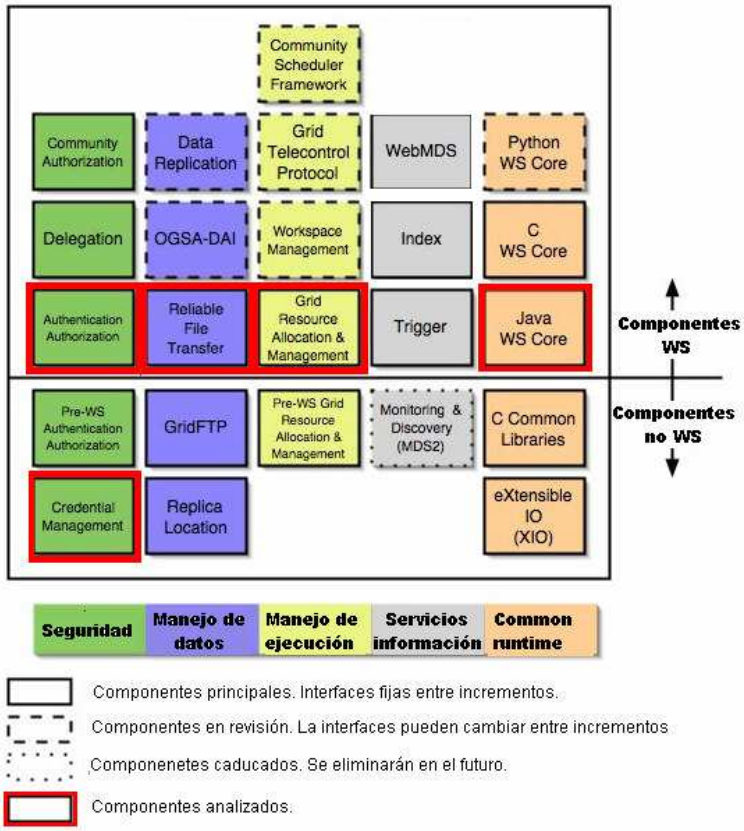

Figura 1.6 Componentes de GT4 analizados. [18]

[2] Este análisis está basado mayormente en la documentación oficial de GT4 encontrada en [18].

Los componentes de GT4, mostrados en la figura 1.6, que se necesitan comprender para el desarrollo de aplicaciones Grid computacionales son:

- Componentes de seguridad.
  - Autenticación y Autorización (Authentication & Authorization).
  - Manejo de Credenciales (Credencial Management).
- Componentes de manejo de datos.
  - Transferencia de Archivos Confiable (Reliable File Transfer).
- Componentes de manejo de ejecución.
  - Manejo y Ubicación de Recursos Grid con Servicios Web (Web Services Grid Resource Allocation & Management).
- Componentes de tiempo de ejecución común.
  - Java WS Core.

1.3.2.1 Componentes de seguridad

GT4 tiene capacidades para: autenticación, autorización, protección de comunicaciones y soporte para las seguridades (manejo de credenciales).

1.3.2.1.1 Autenticación y Autorización (Authentication & Authorization)

GT4 provee mecanismos de seguridad a nivel de mensaje y de transporte.

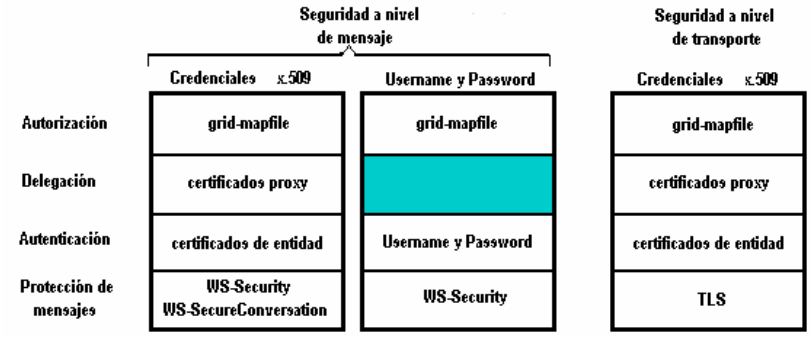

Figura 1.7 Niveles de mecanismos de seguridad.

En la figura 1.7 se muestran los niveles de mecanismos de seguridad. Los componentes de seguridad de GT4 se encargan de autenticación, autorización, delegación y protección de mensajes. Los niveles de

mecanismos de seguridad son:

- Seguridad a nivel de mensaje con credenciales X.509.
- Seguridad a nivel de mensaje con usernames/passwords.
- Seguridad a nivel de transporte con credenciales X.509 (por defecto).

En el cliente la seguridad es manejada programáticamente, mediante llamadas de funciones.
En el servidor la seguridad se maneja declarativamente, mediante un descriptor de seguridad.

Autenticación mutua

Para establecer la comunicación entre entidades[3] primero se produce la autenticación mutua, consistente en que dos entidades se autentiquen la una a la otra.

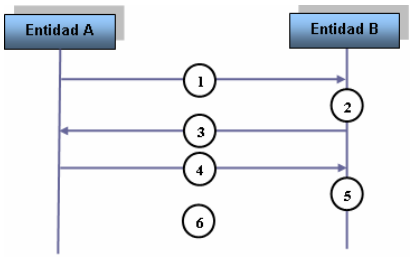

Figura 1.8 Proceso de autenticación mutua.

En la figura 1.8 se muestra el proceso de autenticación mutua.

- En el paso 1, A establece una conexión con B y le da su certificado.
- En el paso 2, B chequea la firma digital de la autoridad certificadora del certificado de A, para esto debe confiar en dicha autoridad certificadora (tener una copia de su certificado).
- En el paso 3, B genera un mensaje randómico y lo envía a A.
- En el paso 4, A encripta el mensaje con su clave privada y lo envía a B.
- En el paso 5, B desencripta el mensaje usando la clave pública de A y si coincide con el mensaje randómico original B sabe que A es auténtico.

[3] Entidades: En este contexto son servicios o usuarios de GT4.

- En el paso 6, ocurren las mismas operaciones a la inversa para que A sepa que B es auténtico.

Comunicación entre entidades

Una vez realizada la autenticación mutua, GT4 permite:

- Comunicación confidencial. Se encriptan los mensajes intercambiados, por lo que se produce mucho overhead en la comunicación.
- Integridad de la comunicación. Por defecto. La comunicación puede ser leída pero no encriptada, produciendo poco overhead.

Claves privadas

GT4 permite almacenar la clave privada del usuario en un archivo de forma encriptada mediante un password.

Certificados proxy

Un certificado proxy es un nuevo certificado y clave privada. Contiene la identidad del dueño modificada para indicar que es un proxy. El nuevo certificado es firmado por el dueño como autoridad certificadora.
Los proxies tienen duración limitada y su clave privada no es encriptada.
Una vez que se ha creado y almacenado el proxy, el usuario puede usarlo para autenticación mutua sin ingresar su password.

Delegación

Es un mecanismo que reduce el número de veces que el usuario debe ingresar su password, mediante el uso de certificados proxy. Esto es útil si se requiere que varios recursos sean usados (cada uno requiriendo autenticación mutua) o si hay solicitud de servicios en nombre de un usuario.

1.3.2.1.2 Manejo de Credenciales (Credential Management)

GT4 provee:

- SimpleCA.
- MyProxy.

a. SimpleCA

Es un paquete que provee una autoridad certificadora simplificada para proveer credenciales a los usuarios y servicios de Globus Toolkit.

b. MyProxy

Es un repositorio de credenciales proxy y de entidad. Se puede guardar credenciales X.509 protegidas por passwords para recuperación posterior por red; esto elimina la necesidad de copiar archivos de certificados y claves privadas entre máquinas.
Se puede establecer políticas de control de acceso a las credenciales en el repositorio.

Este componente tiene herramientas de línea de comandos mayormente para guardar y recuperar credenciales.

1.3.2.2 Componentes de manejo de datos

Los componentes de manejo de datos tienen que ver con la ubicación, transferencia y manejo de datos distribuidos. Existen componentes para: movimiento de datos (GridFTP, RFT) y replicación de datos (Replica Location Service).

1.3.2.2.1 Transferencia de Archivos Confiable (Reliable File Transfer)

GridFTP es un protocolo que provee transferencia de datos segura, robusta, rápida y eficiente. GT4 provee la implementación de un servidor, globus-gridftp-server, y de un cliente, globus-url-copy.

GridFTP no se basa en servicios Web y requiere que el cliente mantenga un socket abierto para la transferencia.

RFT es un servicio basado en WSRF que provee interfaces para control y monitoreo de transferencias de archivos de terceros usando servidores GridFTP. Su principal propósito es guardar el estado de la transferencia. Es esencialmente una versión recuperable y confiable de globus-url-copy.

RFT provee capacidades para:

- Calendarización de tareas de transferencia de datos.
- Monitoreo de transferencia.
- Borrado de archivos y directorios.
- Parámetros (permisos, número de transferencias, etc.) de transferencia.

RFT tiene herramientas de línea de comandos para enviado y monitoreo de transferencias y borrado de archivos.

Mecanismo de transferencia

Para las transferencias se le da una lista de URLs de fuente y destino al servicio y este escribe la descripción de la tarea y mueve los archivos. Se crea un recurso RFT de los archivos a ser transferidos enviando una petición de transferencia (un conjunto de transferencias GridFTP) al servicio Factory RFT. El recurso creado expone el estado de la transferencia como una propiedad de recurso.

El monitoreo de transferencia se realiza mediante mecanismos WSRF estándar (consulta, subscripción/notificación).

Se provee una implementación del servicio, un cliente y clases java (en desarrollo).

1.3.2.3 Componentes de manejo de ejecución

Los componentes de manejo de ejecución permiten la iniciación, monitoreo, administración, calendarización y coordinación de tareas computacionales remotas (jobs) sobre recursos computacionales Grid.

El servidor WS GRAM es comúnmente deployado junto con Delegation (para delegar credenciales para uso de WS GRAM o RFT) y RFT (para ubicación de archivos antes y después de la ejecución); así, se maneja de forma integrada: el monitoreo y administración de la computación, la delegación de credenciales proxies, y la administración de datos.

1.3.2.3.1 Manejo y Ubicación de Recursos Grid con Servicios Web (WS GRAM).

WS GRAM es un conjunto de clientes y servicios Web WSRF, diseñados para el ambiente de hosting de GT4, para comunicación entre calendarizadores usando un protocolo común. WS GRAM no es un calendarizador de recursos, sino que permite la comunicación con calendarizadores de recursos locales. Los servicios son:

a. ManagedJob

Cada tarea enviada es expuesta como un recurso acorde al servicio genérico ManagedJob. El servicio provee una interface para monitorear el estado de la tarea o terminar con la tarea (terminando el recurso ManagedJob).

b. ManagedJobFactory

Cada elemento de cómputo, accedido a través de un calendarizador local, es expuesto como un recurso acorde al servicio genérico ManagedJobFactory. El servicio provee una interface para crear recursos ManagedJob para ejecutar una tarea en ese calendarizador local.

Contexto de WS GRAM

WS GRAM combina servicios de manejo de tareas (jobs) y adaptadores de sistema local con otros componentes de GT4 (RFT, Delegation), para soportar la ejecución de tareas con ubicación de archivos coordinada.

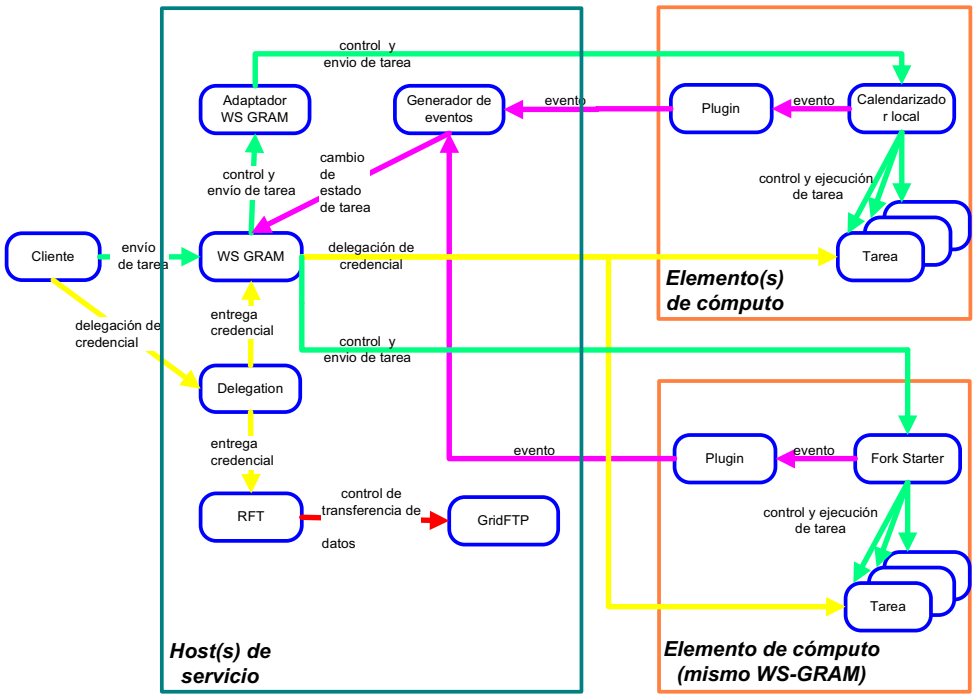

Figura 1.9 Contexto de WS GRAM.

En la figura 1.9 se muestra el contexto de la relación estática de una deployación de WS GRAM. A continuación se describen sus elementos no vistos con anterioridad:

- Calendarizador local de tareas: se puede tener un calendarizador de tareas local, por lo general para elementos de cómputo de gran escala, para administrar los recursos del elemento de cómputo.
- Fork Starter: inicia y monitorea procesos en el mismo servidor WS-GRAM. Ejecuta la tarea y espera que termine. Registra el tiempo de inicio, el tiempo de terminación y el estado de terminación.
- Adaptadores de calendarizadores: El soporte para controlar el calendarizador local es provisto por los adaptadores de acuerdo al API de adaptadores GRAM.

- Generador de eventos de calendarizador (Scheduler Event Generator): Es un componente que provee a WS GRAM la capacidad de monitoreo de tareas.
- Plugin de generador de eventos de calendarizador: provee una interface entre el generador de eventos de calendarizador y los calendarizadores locales.

Secuencia de actividades

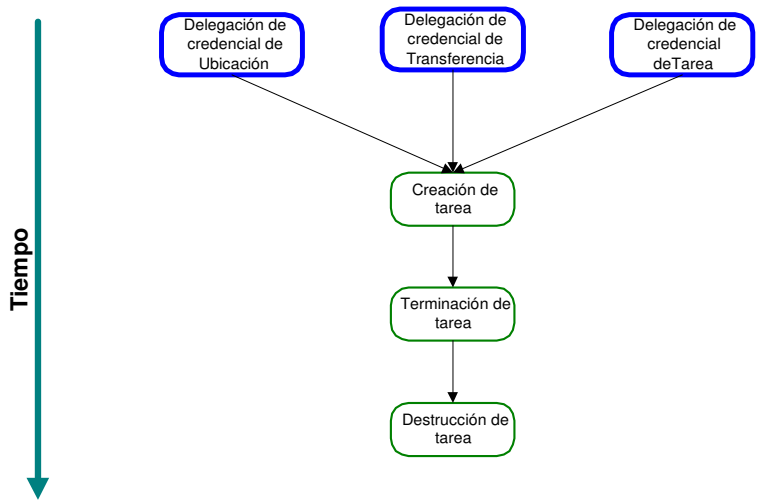

Figura 1.10 Secuencia de actividades.

En la figura 1.10 se muestra la secuencia de actividades concerniente al manejo de tareas con WS GRAM.

- Creación de la tarea: Un cliente WS GRAM crea una tarea (El recurso ManagedJob es creado por el ManagedJobFactory) que seguirá un ciclo de vida donde eventualmente se completa la ejecución y se destruye el recurso de la tarea.
- Delegación de credenciales de ubicación y transferencia: El cliente podría pedir actividades de ubicación para antes (stagein) o después (stageout) de la tarea por parte de WS GRAM. Si estas son pedidas en la creación, se deben pasar EPRs (End Point References) de credenciales delegadas para que las operaciones de delegación se hagan antes de la creación del ManagedJob. Se pasan una credencial para la ubicación (para

interactuar con RFT) y una credencial para transferencias (para interactuar con servidores GridFTP).

- Delegación de credenciales de tarea: El cliente podría pedir que se guarde una credencial en la cuenta del usuario para que el proceso de la tarea la use. Si estas son pedidas en la creación, se deben pasar EPRs de credenciales delegadas para que las operaciones de delegación se hagan antes de la creación del ManagedJob.

1.3.2.4 Componentes de tiempo de ejecución común

GT4 provee facilidades para desarrollar y albergar servicios Grid escritos en Java, C o Python. Los servicios Grid desarrollados pueden utilizar los servicios de infraestructura del Grid.

1.3.2.4.1 Java WS Core

Java WS Core provee:

- APIs y herramientas para desarrollar servicios WSRF.
- Un contenedor de servicios WSRF.
- Herramientas de línea de comandos para manejo de recursos y del contenedor

1.4 METODOLOGÍA DE DESARROLLO

En este punto se presentan los parámetros de selección del problema a resolver, la descripción del problema y la metodología de desarrollo usada para la aplicación.

1.4.1 PARÁMETROS DE SELECCIÓN[4]

Los problemas susceptibles de la aplicación de la técnica de descomposición

[4] Este punto está basado mayormente en la documentación encontrada en [3].

paralela de datos con poco acoplamiento entre los elementos de datos son apropiados para computación Grid, ya que son fácilmente paralelizables y altamente escalables.

La técnica de descomposición paralela de datos es una técnica de descomposición para construir aplicaciones distribuidas en la cual el mismo algoritmo es aplicado a todos los elementos de datos en la aplicación, es más efectivo cuando hay poco acoplamiento entre los elementos. Cuando existe mayor acoplamiento entre elementos este tipo de descomposición es un reto y podría ser impractico. [3 Pág.59]

Con el objetivo de encontrar un problema susceptible de la aplicación de la técnica de descomposición paralela de datos definimos los siguientes parámetros:

Datos

- El conjunto de datos debe poder ser dividido en elementos de datos independientes entre sí (elementos de datos totalmente desacoplados). Así, cada elemento de datos puede ser procesado independientemente.

Procesamiento

- El mismo algoritmo debe aplicarse a todos los elementos de datos. Al no ser el algoritmo dependiente de los datos la resolución del problema se simplifica.
- El procesamiento debe poder ser dividido en tareas independientes entre sí. Adicionalmente a tener los elementos de datos totalmente desacoplados no debe haber coordinación de procesos. Así, la resolución del problema es altamente escalable.

Otros parámetros

Adicionalmente a que el problema sea susceptible de la aplicación de la técnica de descomposición paralela de datos:

- El problema debe requerir mucho cómputo. Esto hará que valga la pena usar Grid computing.
- Los posibles retornos de los procesos deben ser los datos resultantes o la señal de falla. Así, se evita el monitoreo de procesos simplificando la aplicación Grid y el Grid.
- El tipo de ejecución debe ser batch, no interactiva. Así se simplifica el diseño de la lógica de la aplicación.

1.4.2 PROBLEMA SELECCIONADO

El problema seleccionado consiste en clasificar una imagen multiespectral de manera distribuida. Está basado en un problema expuesto en la cita bilbiográfica [21], usado para probar el potencial del sistema de computación paralela PiCEIS (Parallel Computational Environment for Imaging Science).

1.4.2.1 Conceptos[5]

Para el entendimiento del problema a resolver detallamos los siguientes conceptos.

Imagen multiespectral es una imagen ópticamente adquirida en más de un espectro o intervalo de longitud de onda. Está formada por un conjunto de **índices de reflexión** en diferentes **bandas** de frecuencia de un conjunto de áreas determinadas. Una de esas áreas se denomina **píxel**, unidad mínima de información de la imagen y el conjunto de índices de reflexión del píxel se denomina **vector del pixel**. Entonces, cada píxel de la imagen tiene valores numéricos correspondientes a la radiancia recibida por el sensor en cada banda del espectro soportada por el sensor.

Clasificar una imagen multiespectral es calcular la correspondencia de cada uno de los píxeles de una imagen multiespectral a una **clase** (en nuestro caso, conjunto de píxeles identificados como de un mismo tipo, por

[5] Estos conceptos están basados en la documentación encontrada en [25], [26] y [27]

ejemplo “trigo”, “sector urbano”, “granja”) en base a un criterio de similitud.

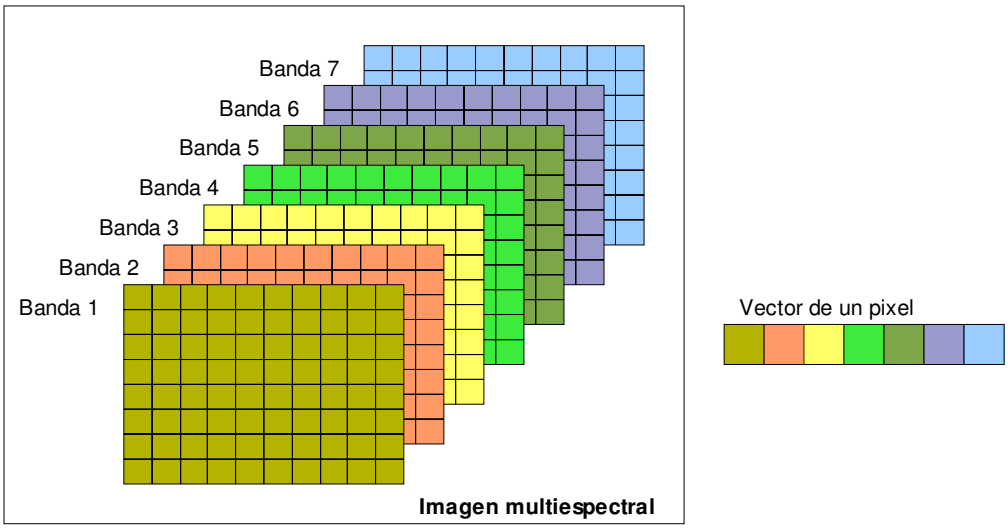

Figura 1.11 Imagen multiespectral.

1.4.2.2 Descripción del problema

El problema consiste en clasificar una imagen multiespectral de manera distribuida. Dado un conjunto de clases (con sus respectivos vectores de píxeles) y una imagen multiespectral, se determinará la pertenencia de cada píxel de la imagen a una de las clases en base a una comparación exhaustiva.

La clasificación se puede dividir en tareas independientes ya que la clasificación de cada píxel es independiente.

1.4.2.3 Justificación de la selección del problema.

Este problema fue sugerido por la Ing. Myriam Hernández, coordinadora general de UNISIG (Unidad de Inteligencia Artificial y Sistemas de Información Geográfica), como un problema que requiere mucho cómputo. Además, ya fue usado para probar el potencial de un sistema de computación paralela por requerir mucho cómputo y ser fácilmente paralelizable.

CLIRSEN, Centro de Levantamientos Integrados de Recursos Naturales por Sensores Remotos, dará asesoría y proporcionará imágenes multiespectrales para desarrollar y probar el funcionamiento de la aplicación. CLIRSEN tiene como misión generar geoinformación y proporcionar servicios técnicos relacionados con los Recursos Naturales y Ambiente, mediante la explotación de las técnicas de Teledetección y Sistemas de Información Geográfica.

En cuanto a los parámetros de selección fijados:

- El conjunto de datos es dividido en elementos de datos independientes entre sí (elementos de datos totalmente desacoplados). Los segmentos de la imagen son independientes entre si.
- El mismo algoritmo se aplica a todos los elementos de datos. El mismo método de clasificación es aplicado a todos los segmentos.
- El procesamiento es dividido en tareas independientes entre sí. La clasificación de cada segmento es independiente.
- Los posibles retornos de cada tarea son el segmento clasificado o la señal de falla.
- El tipo de ejecución es batch. La clasificación de toda una imagen no requiere de interacción con el usuario.

1.4.2.4 Descripción de la aplicación a ser desarrollada

La aplicación a desarrollarse es un clasificador de imágenes multiespectrales, CAMILA.

CAMILA usará las capacidades Grid de GT4 de distribución de procesamiento en su método de clasificación de imágenes. Esta es la parte principal de la aplicación.

Adicionalmente CAMILA deberá dar facilidades para definir las clases de

píxeles requeridas para el proceso de clasificación. Esta es la parte utilitaria.

CAMILA deberá ser una aplicación de escritorio debido a la naturaleza gráfica de la parte utilitaria.

1.4.3 SELECCIÓN DE LA METODOLOGÍA DE DESARROLLO

Para el desarrollo de CAMILA, la hemos separado en: la lógica de la aplicación (mayormente la parte utilitaria) y el método de clasificación de imágenes, debido a que su proceso de desarrollo es muy diferente. Para la lógica de CAMILA se requiere definir requerimientos mientras que el método de clasificación está casi completamente dado.

1.4.3.1 Metodología de desarrollo de la lógica de la aplicación

Para el desarrollo de CAMILA usaremos XP[6], metodología de desarrollo ligera enfocada en la naturaleza cambiante de los requerimientos.
XP es recomendado cuando[7]: los requerimientos del problema son cambiantes; el riesgo del proyecto es alto debido a que el grupo de desarrollo no está familiarizado con el tema; el grupo de desarrollo es pequeño; en el desarrollo del proyecto, trabajan juntos el usuario y los desarrolladores.

XP fue seleccionado debido a que:

1. La aplicación tiene como objetivo demostrar la comprensión conceptual de Grid Computing y del uso de GT4 en la distribución de procesamiento. Razón por la cual: (a) el equipo de desarrollo será considerado el usuario. Lo cual hará más flexible el desarrollo, enfocándonos en el uso de la tecnología y no en la satisfacción del usuario. (b) los requerimientos serán muy dinámicos, según se vea la necesidad de ciertos requerimientos para cumplir con este objetivo, estos pueden ser alterados, removidos y añadidos.

[6] En adelante XP se usará como abreviación de eXtreme Programming.

[7] Este punto está basado mayormente en la documentación encontrada en [24]

2. El equipo de desarrollo no tiene experiencia con los APIs ni la lógica de la aplicación. Razón por la cual: (a) no es realista llevar un calendario. (b) no es realista calcular los esfuerzos. (b) los requerimientos serán muy dinámicos, según se vea la factibilidad de implementar ciertos requerimientos, estos pueden ser alterados, removidos y añadidos.
3. El equipo de desarrollo lo conformamos 2 personas con comunicación constante.

De XP usaremos las siguientes prácticas:

- Programación en parejas. El código será revisado y discutido mientras se escribe.
- Historias del usuario. Las especificaciones de las funcionalidades se harán con la profundidad necesaria para poder realizarla.
- Prototipeo rápido. De manera iterativa e incremental se harán pequeñas mejoras unas tras otras.
- Pruebas de aceptación. Se verificará que se cumpla con las historias del usuario.

De XP no usaremos las siguientes prácticas:

- Calendarización de entregas. Debido a que el equipo de desarrollo es el mismo usuario.
- Metáfora del diseño del sistema. Esta sirve para una fácil introducción de nuevos miembros al equipo, en nuestro caso esto no sucederá.
- Pruebas de unidad formales. Se irá probando el código a medida que este se escriba.

1.4.3.2 Metodología de desarrollo del método de clasificación

Para el método de clasificación definiremos sus requerimientos de computación, diseñaremos la distribución de la computación e instalaremos el ambiente distribuido de computación para soportarlo.

CAPÍTULO II

ANÁLISIS DE REQUERIMIENTOS DE LA APLICACIÓN

En esta sección se presentan los requerimientos de la lógica de la aplicación, los requerimientos de computación del método de clasificación de imágenes multiespectrales y los requerimientos de seguridades.

2.1 REQUERIMIENTOS DE LA LÓGICA DE LA APLICACIÓN

En este punto se presentan las historias del usuario y las restricciones de CAMILA.

Debido a que usamos XP, el desarrollo se lo hizo de manera iterativa e incremental. Lo que incluimos en esta sección no son requerimientos definidos en un inicio sino un compendio de los requerimientos definidos a lo largo del desarrollo de CAMILA.

2.1.1 HISTORIAS DEL USUARIO

Por cada requerimiento de usuario de CAMILA hay una historia del usuario. Las historias del usuario tienen la profundidad necesaria para poder entender la funcionalidad; no se consideran tecnologías ni restricciones técnicas. Están escritas en lenguaje natural, no técnico usando una o dos frases. No tendrán en cuenta tiempos debido a que no se calendarizarán las entregas.

CAMILA permitirá:

- **Abrir una imagen multiespectral.**

Graficar una imagen, sobre la cual se trabajará, a partir de los archivos de las bandas de la imagen ubicados por el usuario. Se usarán las bandas 3, 5 y 7 para los colores rojo, verde y azul respectivamente.

- **Componer la imagen multiespectral.**

  Graficar la imagen abierta a partir de bandas escogidas por el usuario para los colores rojo, verde y azul.

- **Ampliar la imagen multiespectral.**

  Aumentar el tamaño de la imagen abierta. Esto permitirá una mayor precisión en la selección de píxeles a añadir a una clase.

- **Reducir la imagen multiespectral ampliada.**

  Retornar la imagen ampliada a su tamaño normal.

- **Crear clase de píxeles.**

  Definir un nombre y un color asociados a un conjunto de píxeles inicialmente vacío.

- **Añadir píxeles a una clase.**

  Seleccionar píxeles de la imagen multiespectral y añadirlos a una clase.

- **Clasificar la imagen multiespectral.**

  A partir de las clases aplicar el algoritmo de clasificación distribuida a la imagen multiespectral y graficar el resultado.

- **Cancelar la clasificación.**

  Cancelar la clasificación que se está ejecutando.

2.1.2 RESTRICCIONES

Sobre el tipo de imágenes con las que trabajará CAMILA:

- Están en formato BSQ (band sequential) en archivos separados por banda. Cada archivo contiene los índices de reflexión de todos los píxeles en la banda correspondiente, uno detrás de otro, ocupando un byte cada uno. Así, la imagen está repartida en 7 archivos. Es decir,

los índices de reflexión de las bandas de un píxel están repartidos en los 7 archivos.

- Pertenecen al sensor TM (thematic mapper) del satélite LANDSAT 5. La especificación de estas imágenes se describe en la tabla 2.1.

Banda	Región del espectro electromagnético	Resolución
1	Visible (azul-verde) (0.45 – 0.52µm)	30 X 30 metros
2	Visible (verde)(0.52 – 0.60µm)	30 X 30 metros
3	Visible (rojo)(0.63 – 0.69µm)	30 X 30 metros
4	Infrarojo (0.76 – 0.90µm)	30 X 30 metros
5	Infrarojo (1.55 – 1.75µm)	30 X 30 metros
6	Infrarojo (10.40 – 12.50µm).	120 X 120 metros
7	Infrarojo (2.08 – 2.35µm)	30 X 30 metros

Tabla 2.1 Especificación de imágenes del sensor TM-LANDSAT 5

CAMILA accederá a los archivos de las imágenes mediante el sistema de archivos local.

CAMILA trabajara sobre una imagen abierta a la vez.

CAMILA será desarrollada para un computador con las siguientes características:

- Procesador Pentium 4 de 3.2 GHz.
- 1 GB de memoria RAM.
- Sistema operativo Debian Sarge.
- Ambiente de escritorio GNOME.

Camila requiere:

- Sistema operativo GNU/Linux con GTK2
- Máquina virtual Java 5.
- 512 MB de RAM libres. Necesario para el manejo estructurado de las imágenes que en disco tienen tamaños desde 70 MB.

2.2 REQUERIMIENTOS DE COMPUTACIÓN

En este punto se define el método de clasificación de imágenes multiespectrales y la computación que se requiere distribuir sin considerar tecnologías.

2.2.1 MÉTODO DE CLASIFICACIÓN DE IMÁGENES MULTIESPECTRALES

Para el entendimiento de qué computación se requiere distribuir es necesario definir el método de clasificación de imágenes multiespectrales.

El método de clasificación de imágenes multiespectrales, según lo expuesto en la sección 1.4.2.2 “Descripción del problema”, se define de la siguiente manera:

Se compara el vector de cada píxel con un conjunto de vectores que han sido previamente identificados como pertenecientes a cierta clase. El algoritmo calcula la distancia de cada píxel de la imagen multiespectral con respecto a cada píxel de la clase. Estas distancias son ordenadas de menor a mayor y a partir de un percentil dado se obtiene la distancia del píxel de la imagen multiespectral a la clase. Se hace esto para cada clase. El píxel pertenece a la clase con la que tiene menor distancia.

La distancia entre dos píxeles es la sumatoria de las diferencias entre sus correspondientes índices de bandas.

Este método de clasificación permite dividir el procesamiento en varias tareas independientes. Cada tarea consiste en clasificar un segmento de la imagen.

2.2.2 COMPUTACIÓN A DISTRIBUIRSE

En este punto se explica qué computación se requiere distribuir.

Se dividirá la imagen en segmentos a ser clasificados en tareas independientes según el método de clasificación definido en la sección 2.2.1 “Método de clasificación de imágenes multiespectrales”. Cada tarea será un proceso independiente.

El número de tareas dependerá del tamaño de la imagen y del tamaño del segmento.

La cantidad de computación a distribuirse es igual al número de operaciones involucradas en la clasificación que serán distribuidas. Este valor depende del tamaño de la imagen y del tamaño y número de clases involucradas.

2.3 REQUERIMIENTOS DE SEGURIDADES

Los mecanismos de seguridad usados en una aplicación Grid deben estar acorde a los usados en el Grid.

Conceptualmente un Grid real está formado por instituciones autónomas que al hacer uso del Grid exponen datos y comunicaciones que son solo de su interés en un ambiente interinstitucional. Esto hace que se deba soportar autenticación, autorización y seguridades en la comunicación.

Debido a que es común en un Grid real, hemos decidido que:

- CAMILA usará facilidades de GT4 para autenticación.
- El Grid autenticará y autorizará al usuario del Grid.
- Se usará encriptación en la comunicación mediante encriptación de clave pública.

CAPÍTULO III

DISEÑO DE LA APLICACIÓN

En esta sección se presenta el diseño de la lógica de CAMILA, la distribución de la computación del método de clasificación de imágenes multiespectrales y el diseño de seguridades.

3.1. DISEÑO DE LA LÓGICA DE LA APLICACIÓN

En este punto se definen las tecnologías, el diseño de las interfaces gráficas y de las funcionalidades a implementar relativas a la lógica de CAMILA.

Debido a que usamos XP, el desarrollo se lo hizo de manera iterativa e incremental. El diseño se ha ido haciendo según lo que planeamos en cada iteración. XP no define una etapa de diseño, hay pasos de diseño para la implementación de lo planeado en cada iteración. Cada iteración es incremental y/o de refactoring (proceso de mejora del diseño del código y de su implementación). El diseño al que se refiere XP es relativo a cada tarea de ingeniería, hecha a partir de una historia del usuario.

En esta sección lo que incluimos no es el diseño al que se refiere XP sino el diseño, no usado en el desarrollo, simplificado de la aplicación final con el fin de facilitar la comprensión de CAMILA.

3.1.1 TECNOLOGÍAS

Las tecnologías usadas son:

El lenguaje de programación Java. Debido a que es el lenguaje que mejor soporta GT4. Adicionalmente, GT4 soporta C y Python.

La plataforma gráfica RCP (Rich Client Platform). Debido a que maneja de forma integrada el look-and-feel de la aplicación. Se ha escogido RCP debido a que trabaja con java, ofrece un look-and-feel nativo y es software libre.

El IDE Eclipse. Debido a que da facilidades para la construcción de aplicaciones basadas en RCP y es software libre.

3.1.2 DISEÑO DE INTERFACES

Las desiciones de diseño de interfaces se han hecho en base a las posibilidades de RCP debido a que se la usará como plataforma gráfica para la implementación de CAMILA.

Gráficamente la aplicación tiene las secciones mostradas en la figura 3.1:

Figura 3.1 Secciones de la aplicación.

- Sección de menú

  Escogiendo ítems del menú se ejecutarán wizards y acciones.
- Sección de barra de herramientas

  Escogiendo los iconos se ejecutarán wizards y acciones. Los iconos tienen un item de menú correspondiente en la sección de menú.
- Sección de presentación de imágenes

Estará dividida en dos partes. En la parte izquierda se visualizará la imagen multiespectral compuesta y en la parte derecha se visualizará la imagen clasificada. Estas partes podrán ampliarse y reducirse de manera que ocupen hasta toda la sección de presentación de imágenes.

3.1.3 DISEÑO DE FUNCIONALIDADES

En este punto se explican las funcionalidades en mayor detalle a partir de las historias del usuario fijadas en 2.1.1 tomando en cuenta las tecnologías usadas. Están escritas en lenguaje no técnico con la profundidad necesaria para que el siguiente paso sea entender la implementación de la funcionalidad.

Lo expuesto en este punto no fue usado para las tareas de ingeniería, sino que tiene el propósito de facilitar la comprensión de la implementación de las funcionalidades.

La mayor parte de las funcionalidades de CAMILA se hacen mediante wizards RCP, el resto se harán mediante acciones RCP.

Las funcionalidades se explican a continuación:

Funcionalidad	**Abrir una imagen multiespectral**
Descripción	Se graficará una imagen a partir de los archivos de las bandas de la imagen ubicados por el usuario. Se usarán las bandas 3, 5 y 7 para los colores rojo, verde y azul respectivamente. Los datos de las bandas se usarán como grados de intensidad de los colores.
Implementación	Esta funcionalidad se implementará con el wizard “Open multispectral image”.
Restricciones	• Dependiendo del tamaño se dividirá a la imagen en partes a ser cargadas. Esto debido a las limitaciones de memoria de la máquina en que

	corre CAMILA. El tamaño de imagen o parte de imagen y el número de partes serán determinadas posteriormente de manera empírica. • No se deberá permitir abrir una imagen multiespectral si: o Se está abriendo otra imagen. o Se está componiendo una imagen. o Se está clasificando una imagen.

Tabla 3.1 Descripción de "Abrir una imagen multiespectral".

Funcionalidad	**Componer la imagen multiespectral**
Descripción	Se graficará la imagen o parte de imagen, abierta con el wizard anterior, a partir de bandas escogidas por el usuario para los colores rojo, verde y azul. Los datos de las bandas se usarán como grados de intensidad de los colores.
Implementación	Esta funcionalidad se implementará con el wizard "Composite multispectral image".
Restricciones	• Si la imagen se encuentra ampliada, al hacer la composición esta se hará sobre toda la imagen o parte de imagen abierta. • No se deberá permitir componer una imagen si: o No hay una imagen abierta. o Se está abriendo otra imagen. o Se está componiendo una imagen. o Se está clasificando una imagen.

Tabla 3.2 Descripción de "Componer una imagen multiespectral".

Funcionalidad	**Ampliar la imagen multiespectral**
Descripción	Se graficará de manera ampliada una sección de la imagen escogida por el usuario.
Implementación	Esta funcionalidad se implementará con la acción "Zoom

	in”.
Restricciones	• No se hará una ampliación de toda la imagen o parte de imagen abierta debido a las limitaciones de memoria de la máquina en que corre CAMILA y el tiempo que tomaría ampliar toda la imagen o parte de imagen. • Solo habrá un factor de ampliación. • El factor de ampliación será determinado posteriormente de manera empírica. • No se deberá permitir ampliar una imagen si: o No se ha seleccionado una sección de la imagen. o La imagen ya está ampliada. o Se está abriendo otra imagen. o Se está componiendo una imagen. o Se está clasificando una imagen.

Tabla 3.3 Descripción de “Ampliar la imagen multiespectral”.

Funcionalidad	**Reducir la imagen multiespectral ampliada**
Descripción	Se graficará nuevamente la imagen original.
Implementación	Esta funcionalidad se implementará con la acción “Zoom out”.
Restricciones	• No se deberá permitir reducir una imagen si: o No existe una imagen ampliada. o Se está abriendo otra imagen. o Se está componiendo una imagen. o Se está clasificando una imagen.

Tabla 3.4 Descripción de “Reducir la imagen multiespectral ampliada”.

Funcionalidad	**Crear clase de píxeles**
Descripción	Se definirá por parte del usuario un nombre y un color para la clase a crearse.

Implementación	Esta funcionalidad se implementará con el wizard "Create new class".
Restricciones	Ninguna clase deberá tener nombre o color repetido.

Tabla 3.5 Descripción de "Crear clase de píxeles".

Funcionalidad	**Añadir píxeles a una clase**
Descripción	Se añadirán los píxeles de la imagen multiespectral seleccionados por el usuario a una clase escogida.
Implementación	Esta funcionalidad se implementará con el wizard "Add pixels to class".

Restricciones	• No se deberá permitir añadir píxeles repetidos. Esto entorpecería el funcionamiento del algoritmo.
	• No podrán ser añadidos los mismos píxeles a dos clases diferentes.
	• La selección de píxeles se hará mediante el mouse por rectángulos sobre la imagen multiespectral graficada.
	• No se deberá permitir ampliar una imagen si:
	  o No se ha seleccionado una sección de la imagen.
	  o No existe al menos una clase.
	  o Se está abriendo otra imagen.
	  o Se está componiendo una imagen.
	  o Se está clasificando una imagen.

Tabla 3.6 Descripción de "Añadir píxeles a una clase".

Funcionalidad	**Clasificar la imagen multiespectral**
Descripción	Se clasificará toda la imagen correspondiente a la imagen o parte de imagen abierta con el wizard "Open multispectral image". Para esto el usuario proporcionará: el percentil usado para calcular la proximidad de un píxel a una clase y el tamaño de cada segmento a clasificarse

	en una tarea. Se graficará el resultado de la clasificación en la sección de imagen clasificada.
Implementación	Esta funcionalidad se implementará con el wizard "Classify multispectral image".
Restricciones	• No se deberá permitir hacer la clasificación si: o No hay una imagen abierta. o No hay clases. o Alguna clase no tiene píxeles. o Se está ejecutando otra clasificación.

Tabla 3.7 Descripción de "Clasificar la imagen multiespectral".

Funcionalidad	**Cancelar la clasificación**
Descripción	Se cancelará la clasificación que se esté ejecutando.
Implementación	Esta funcionalidad se implementará con la acción "Cancel classification".
Restricciones	• No se deberá permitir cancelar la clasificación si: o No hay una clasificación ejecutándose. o La clasificación está en proceso de cancelación.

Tabla 3.8 Descripción de "Cancelar clasificación".

3.2. DISTRIBUCIÓN DE LA COMPUTACIÓN

En este punto se define el algoritmo de clasificación distribuida de imágenes multiespectrales y el escenario de la distribución de la computación considerando las tecnologías a usarse.

3.2.1 TECNOLOGÍAS A USARSE

Se usarán las librerías del cliente de WS-GRAM. Debido a que usaremos WS-GRAM para la distribución de procesamiento. WS-GRAM es el componente de GT4 que permite el manejo de ejecución.

3.2.2 ALGORITMO DE CLASIFICACIÓN DISTRIBUIDA DE IMÁGENES MULTIESPECTRALES

El algoritmo del método de clasificación de imágenes multiespectrales se define a continuación:

1. Se divide la imagen multiespectral en segmentos.
2. Se clasifica cada segmento de la imagen multiespectral.

   Cada segmento es clasificado en un job de WS-GRAM.

   2.1. Se clasifica cada píxel del segmento.

   2.1.1. Se calcula la distancia entre cada píxel de la imagen y cada clase.

   2.1.1.1. Se calculan las distancias entre el píxel de la imagen y los píxeles de la clase.

   2.1.1.1.1. Se restan los valores de cada banda entre píxeles.

   2.1.1.1.2. Se suman los valores absolutos de las restas.

   Esta es la distancia entre píxeles.

   2.1.1.2. Se ordenan de menor a mayor las distancias.

   2.1.1.3. Se escoge la distancia de mayor valor que se encuentra dentro de un percentil dado.

   Esta es la distancia entre el píxel y la clase

   2.1.2. Se calcula la pertenencia del píxel a una clase.

   El píxel pertenece a la clase cuya distancia entre píxel y clase es la menor.

3. Se grafica cada segmento clasificado

   A partir de la pertenencia a una clase, cada píxel del segmento usará el color de esa clase para graficar la imagen clasificada.

El algoritmo es ilustrado en el siguiente diagrama de actividad.

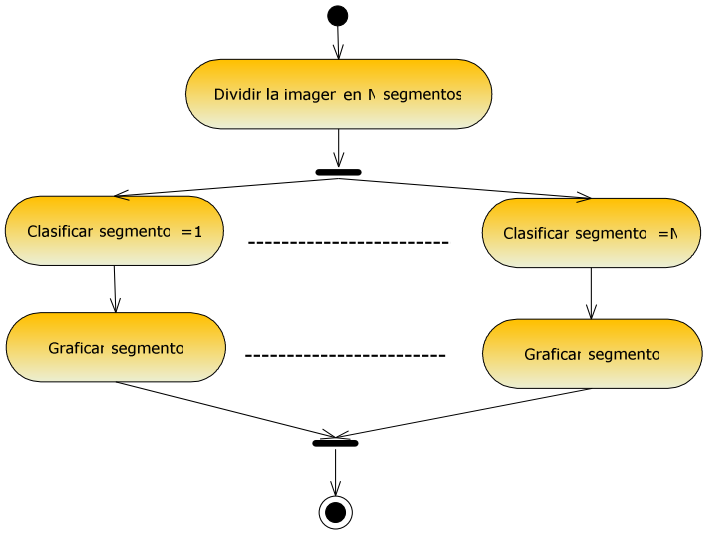

Figura 3.2 Algoritmo de clasificación.

Donde la actividad "Clasificar segmento", que es ejecutada en un job de WS-GRAM, es la siguiente.

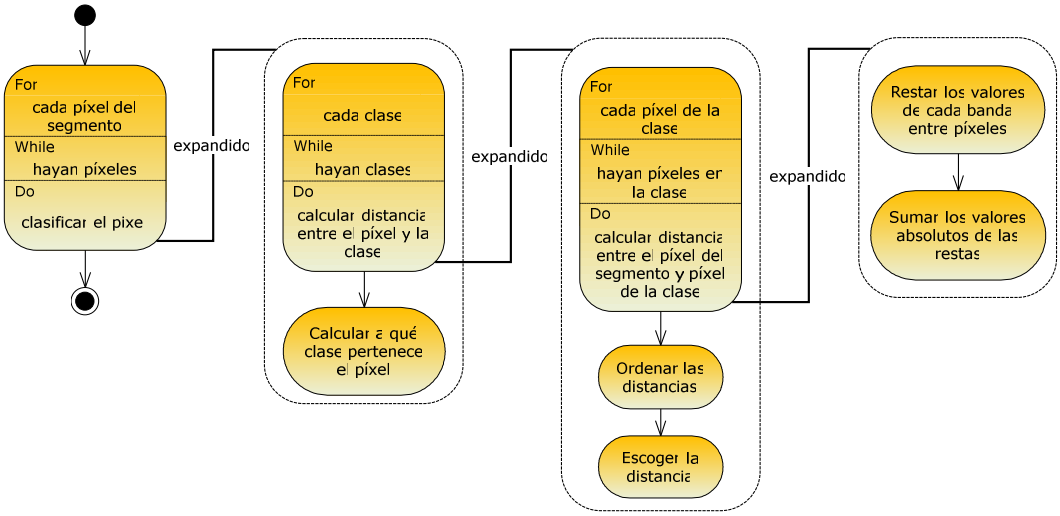

Figura 3.3 Actividad "clasificar segmento".

3.2.3 ESCENARIO DE LA DISTRIBUCIÓN DE LA COMPUTACION

En la sección 2.2.2 "Computación a distribuirse" definimos que la computación que requerimos distribuir es la clasificación de los segmentos.

En este punto describimos cómo se distribuye la computación usando GT4.

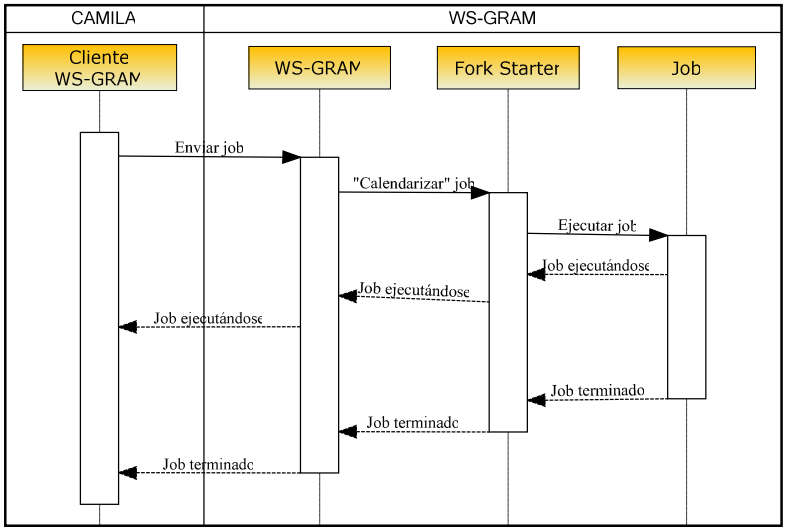

Figura 3.4 Escenario de ejecución[8].

En la figura 3.4 se muestra el escenario de ejecución. Tendremos dos entidades involucradas: CAMILA y un servidor WS-GRAM. CAMILA envía un job al servidor WS-GRAM y recibe notificaciones de que el job se está ejecutando y de que ha terminado. El servidor WS-GRAM usará Fork Starter para ejecutar el job en un proceso en el mismo servidor.

[8] Para mejor entendimiento ver la figura 1.9 “Contexto de WS GRAM” donde se expone un escenario más general.

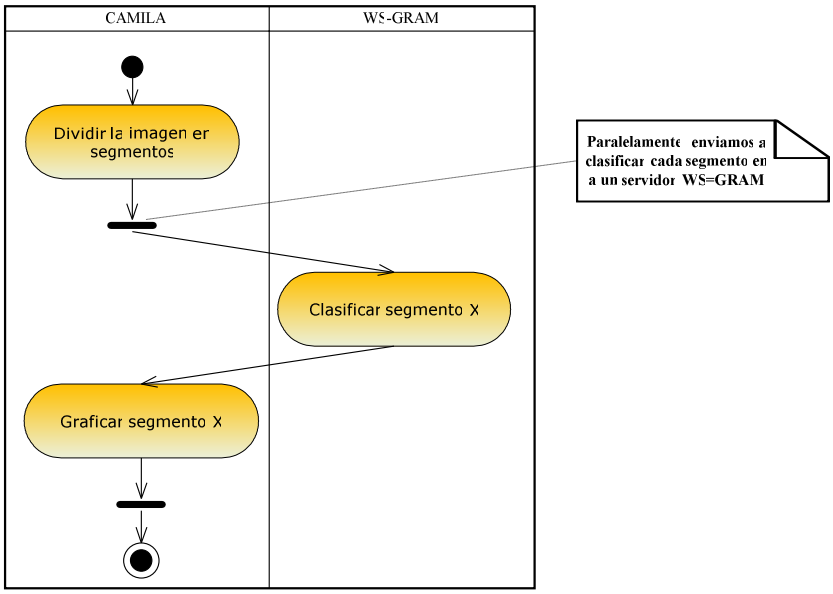

Figura 3.5 Distribución de la clasificación.

En la figura 3.5 se muestra la distribución de la clasificación tomado en cuenta el algoritmo de clasificación, descrito en la figura 3.2, y el escenario de ejecución, descrito en la figura 3.4. CAMILA, usando las librerías del cliente de WS-GRAM, enviará jobs al servidor consistentes en la clasificación de un segmento.

3.3. DISEÑO DE SEGURIDADES

En este punto definimos la manera en que se soportará lo definido en 2.3 “Requerimientos de seguridades”, es decir los mecanismos de seguridad a implementar.

Para la autenticación y la encriptación en la comunicación CAMILA y el Grid usarán certificados X.509. Debido a que GT4 soporta este tipo de certificados.

Tendremos una autoridad certificadora. Esta será el componente SimpleCA. SimpleCA es el componente de GT4 que implementa una autoridad

certificadora.

Para la autorización se usará el “grid map file”. Este archivo contiene mapeos del subject de un certificado de un usuario a usuarios locales del sistema operativo. De esta manera cada usuario debidamente autenticado tendrá procesos en el servidor GT4 bajo el usuario local del sistema operativo.

Para el envío de jobs se usarán certificados proxy del usuario ya que el servidor WS-GRAM realizará tareas en nombre del usuario.

CAPÍTULO IV

IMPLEMENTACIÓN DE LA APLICACIÓN[9]

En esta sección se explica cómo está construida CAMILA, como está instalado el ambiente distribuido de ejecución para soportar el método de clasificación de imágenes multiespectrales y las pruebas de funcionamiento del método distribuido de clasificación de imágenes multiespectrales.

Esta sección está escrita en lenguaje técnico para quien quiera conocer los detalles de la implementación tanto de CAMILA como del Grid.

4.1 IMPLEMENTACIÓN DE LA LÓGICA DE LA APLICACIÓN

En este punto se explican detalles de cómo está construida CAMILA.

Para la implementación de la aplicación se usó el lenguaje JAVA con RCP, las librerías de cliente de WS-GRAM y librerías para uso de servicios WSRF.

4.1.1 CONSIDERACIONES DE MEMORIA

Debido al tamaño de las imágenes y a su representación (estructuras de datos) en CAMILA, fue necesario hacer consideraciones sobre el uso de memoria. Las principales fueron:

- La aplicación arranca usando 512 MB de memoria.

[9] Para el entendimiento de este capitulo es necesario tener conocimientos del lenguaje Java, el framework RCP, GT4 (principalmente WS-GRAM y WSRF) y UML. La información necesaria acerca de GT4 se encuentra en el marco teórico. Se asumen conocimientos de Java y UML. Acerca de RCP, referirse al sitio Web del proyecto Eclipse www.eclipse.org. Ciertos términos de las tecnologías nombradas se encuentran en el glosario.

- En caso de que la imagen sea muy grande se la divide en secciones para poder abrirla.
- Los tipos de datos usados serán los que más ahorren espacio en memoria. Esto implica que deba usarse un tipo de dato byte (cuyos valores están entre –128 y 127) para los valores de índice de reflexión (entre 0 y 255).
- Se usan exclusivamente arreglos para manejar conjuntos. No se usa el framework Collections de java.

4.1.2 EN CUANTO AL FRAMEWORK RCP

La aplicación se hizo usando las facilidades de Eclipse para desarrollo de aplicaciones RCP. Concretamente se creó un “Plug-in project” con las librerías de Eclipse 3.2 para que sea una aplicación aparte del IDE.

Se usan las extensiones:

org.eclipse.core.runtime.applications

org.eclipse.core.runtime.products

org.eclipse.ui.commands

org.eclipse.ui.perspectives

org.eclipse.ui.views

org.eclipse.ui.bindings

La aplicación final es un producto Eclipse exportado a partir de una configuración de producto.

Así como en el diseño, la implementación de CAMILA fue altamente afectada por el uso del framework RCP.

4.1.3 DESCRIPCIÓN ESTÁTICA DE LA APLICACIÓN

Se describen los paquetes y clases[10] Java más importantes de la aplicación.

Las clases organizadas por paquetes son las siguientes:

- Paquete camila.rcp.workbench

  Contiene clases e interfaces usadas para el manejo del ciclo de vida de la aplicación. Son específicas de RCP. Son usadas por el Eclipse Workbench.

- Paquete camila.rcp.perspective

  Contiene solo la clase Perspective.

Clase	Descripción
Perspective	Genera el layout inicial de la ventana de la aplicación. Es específica de RCP. Es usada por el Eclipse Workbench.

Tabla 4.1 Descripción del paquete "camila.rcp.pespective".

- Paquete camila.rcp.view

  Contiene las clases relacionadas con los views usados para desplegar las imágenes. En la sección 3.1.2 "Diseño de interfaces" definimos la sección de presentación de imágenes, la cual está dividida en imagen compuesta e imagen clasificada. Esta sección está implementada mediante dos Vistas RCP.

Clase	Descripción
MultispectralImageView	Define la vista usada para desplegar la imagen multiespectral compuesta. Es específica de RCP. Es usada por el Eclipse Workbench.

[10] La documentación de todas las clases y paquetes generada con el utilitario javadoc, a partir de los comentarios de clases, miembros dato y miembros método, se encuentran en el directorio camila/doc en el producto generado de CAMILA incluido en el CD de esta tesis.

ClassifiedImageView	Define la vista usada para desplegar la imagen clasificada. Es específica de RCP. Es usada por el Eclipse Workbench.

Tabla 4.2 Descripción del paquete "camila.rcp.view".

- Paquete camila.rcp.action

  Contiene clases relacionadas con la definición de acciones RCP. Son específicas de RCP. Son usadas por el Eclipse Workbench.

Clase	Descripción
OpenMultispectralImageAction	Define la acción usada para abrir la imagen multiespectral. Inicia el wizard definido en la clase OpenMultiespectralImageWizard.
CompositeMultispectralImageAction	Define la acción usada para componer la imagen multiespectral. Inicia el wizard definido en la clase CompositeMultiespectralImageWizard
CreateNewClassAction	Define la acción usada para crear una nueva clase de firmas multiespectrales. Inicia el wizard definido en la clase CreateNewClassWizard.
AddPixelsToClassAction	Define la acción usada para añadir píxeles a una clase. Inicia el wizard definido en la clase AddPixelsToClassWizard.
ClassifyMultispectralImageAction	Define la acción usada para clasificar la imagen multiespectral. Inicia el wizard definido en la clase ClassifyMultiespectralImageWizard.

CancelClassificationAction	Define la acción usada para cancelar la clasificación de una imagen multiespectral.
ZoomInMultispectralImageAction	Define la acción usada para ampliar la imagen multiespectral.
ZoomOutMultispectralImageAction	Define la acción usada para reducir la imagen multiespectral ampliada.

Tabla 4.3 Descripción del paquete "camila.rcp.action".

- Paquete camila.rcp.wizard

Contiene las clases relacionadas con los wizards usados. Cada wizard tiene una clase page asociada. Son específicas de RCP. Son usadas por el Eclipse Workbench.

Clase	Descripción
OpenMultispectralImageWizard	Define el wizard usado para abrir la imagen multiespectral. Contiene un objeto OpenMultiespectralImagePage.
CompositeMultispectralImageWizard	Define el wizard usado para componer la imagen multiespectral. Contiene un objeto CompositeMultiespectralImagePage.
CreateNewClassWizard	Define el wizard usado para crear una nueva clase de firmas multiespectrales. Contiene un objeto CreateNewClassPage.
AddPixelsToClassWizard	Define el wizard usado para añadir píxeles a una clase. Contiene un objeto AddPixelsToClassPage.

ClassifyMultispectralImageWizard	Define el wizard usado para clasificar la imagen multiespectral. Contiene un objeto ClassifyMultiespectralImagePage.

Tabla 4.4 Descripción del paquete "camila.rcp.wizard".

- Paquete camila.app.utils

  Contiene clases (instanciables y estáticas), enumerations y exceptions utilitarios usados en el desarrollo de CAMILA.

Clase	Descripción
Band	Es un enumeration. Contiene valores fijos correspondientes a las bandas.
DataTypeConverter	Contiene un método de conversión de datos byte a short. Ya que el lenguaje Java no provee un tipo de dato de 8 bits sin signo (los valores del tipo de dato byte están entre –128 y 127) y debido a las limitaciones de memoria en la forma en que guardamos los valores de índice de reflexión (el cual está entre 0 y 255): usamos el tipo de dato byte para guardar el valor pero su valor no refleja el valor de índice de reflexión. Para usar el valor debe ser convertido a un tipo de dato short (el cual sí permite valores entre 0 y 255).
GramFileManager	Contiene métodos útiles para el manejo de archivos relacionados con WS-GRAM. Maneja los archivos creados y leídos del sistema de archivos local.

HeaderInformation	Contiene información obtenida del archivo header de la imagen multiespectral. Es usada en el retorno del método que lee este archivo de la clase MultispectralFileManager.
IncorrectHeaderFileException	Es una excepción. Es lanzada cuando el formato del archivo del Header es incorrecto.
MultispectralFileManager	Contiene métodos para obtener información de los archivos del header y de las bandas.
SWTUtility	Contiene métodos útiles relacionados con SWT.

Tabla 4.5 Descripción del paquete "camila.app.utils".

- Paquete camila.app.mi

Contiene clases que se encargan del manejo de la información de la imagen multiespectral.

Clase	Descripción
MultispectralImageManager	Maneja toda la información de la imagen multiespectral, esto es: la imagen misma, la parte cargada, la parte gráfica y los archivos de la imagen. Esta clase implementa el patrón facade para sus miembros dato. Esto quiere decir que no se permitirá obtener una referencia a sus miembros dato sino que se usarán métodos.
MultispectralImage	Maneja información de la imagen multiespectral.
MultispectralImagePart	Maneja información de la parte de la imagen

	cargada.
MultispectralImageGraphics	Maneja la información de graficación de la imagen multiespectral. Usa las librerías de RCP.
MultispectralImageFiles	Maneja la información de los archivos de la imagen multiespectral.

Tabla 4.6 Descripción del paquete "camila.app.mi".

- Paquete camila.app.open

Contiene solo la clase OpenMultispectralImageRCPJob.

Clase	Descripción
OpenMultispctralImageRCPJob	Abre una imagen multiespectral usando un job RCP que corre en el background.

Tabla 4.7 Descripción del paquete "camila.app.open".

- Paquete camila.app.composite

Contiene solo la clase CompositeMultispctralImageRCPJob.

Clase	Descripción
CompositeMultispctralImageRCPJob	Compone la imagen multiespectral usando un job RCP que corre en el background.

Tabla 4.8 Descripción del paquete "camila.app.composite".

- camila.app.classes

Contiene solo la clase Perspective.

Clase	Descripción
SignatureClasses	Maneja un conjunto de clases de firmas multiespectrales. Una sola instancia de esta clase es

	usada a través de la aplicación.
SignatureClass	Maneja información acerca de una clase de firmas multiespectrales.

Tabla 4.9 Descripción del paquete "camila.app.clases".

- camila.app.classify.gram

Contiene clases involucradas en la clasificación usando WS-GRAM.

Clase	Descripción
GRAMJobInformation	Maneja información necesaria para crear y manejar un job WS-GRAM.
GRAMInformation	Maneja información acerca de todos los jobs WS-GRAM involucrados en una clasificación.
RSLGenerator	Se encarga de crear un objeto JobDescriptionType que contiene información acerca de un job WS-GRAM a partir de un objeto GRAMJobInformation.
GRAMClassificationManager	Administra una clasificación. Crea los jobs RCP de clasificación.
GRAMClassificationRCPJob	Es un job RPC que corre en el background. Crea un job WS-GRAM y escucha el estado del job WS-GRAM.
GRAMClassification GeneralRCPJobInformation	Mantiene información usada por todos los jobs GRAMClassificationRCPJob. Es el punto de comunicación entre estos jobs.
GRAMClient	Contiene métodos que soportan la creación de jobs WS-GRAM su envío a un servidor WS-GRAM.

GRAMClassifiedImageSegment	Representa un segmento de imagen clasificado recuperado desde disco como resultado de una clasificación.
GRAMResultProcessorRCPJob	Es un job RPC que corre en el background. Básicamente se encarga de graficar los segmentos clasificados.

Tabla 4.10 Descripción del paquete "camila.app.classify.gram".

- Paquete camila.app.ci

Contiene solo la clase ClassifiedImageManager.

Clase	Descripción
ClassifiedImageManager	Maneja la información de la imagen clasificada.

Tabla 4.11 Descripción del paquete "camila.app.ci".

- Paquete camila.app.session

Contiene solo la clase Session.

Clase	Descripción
Session	Maneja la información que debe ser accedida a través de la instancia de la aplicación.

Tabla 4.12 Descripción del paquete "camila.app.session".

- Paquete camila.classifier

Contiene solo la clase MultispectralImageClassifier.

Clase	Descripción
MultispectralImageClassifier	Realiza la clasificación de un segmento de imagen multiespectral en un nodo de ejecución. Está empaquetada en el archivo classifier.jar que se enviará al nodo de

	ejecución.

Tabla 4.13 Descripción del paquete "camila.classifier".

4.1.4 DESCRIPCIÓN DINÁMICA DE LA APLICACIÓN

Se describen las llamadas entre las clases y objetos involucrados en cada historia del usuario mediante diagramas UML de secuencia.

Los diagramas tienen la intención de dar una idea general de la implementación de la aplicación. No describen el código en su totalidad ni profundamente. Estos diagramas facilitan la lectura del código.

Como convención:

- No se detallan todas las llamadas y en ciertos casos las llamadas se abrevian en una sola acción, esto por motivos de entendimiento.
- Los métodos se distinguen por el uso de "()".
- En los casos en que se abrevien las llamadas se usará una frase explicativa sin "()".
- Solo en casos en que se mejore el entendimiento se usarán argumentos y retornos.

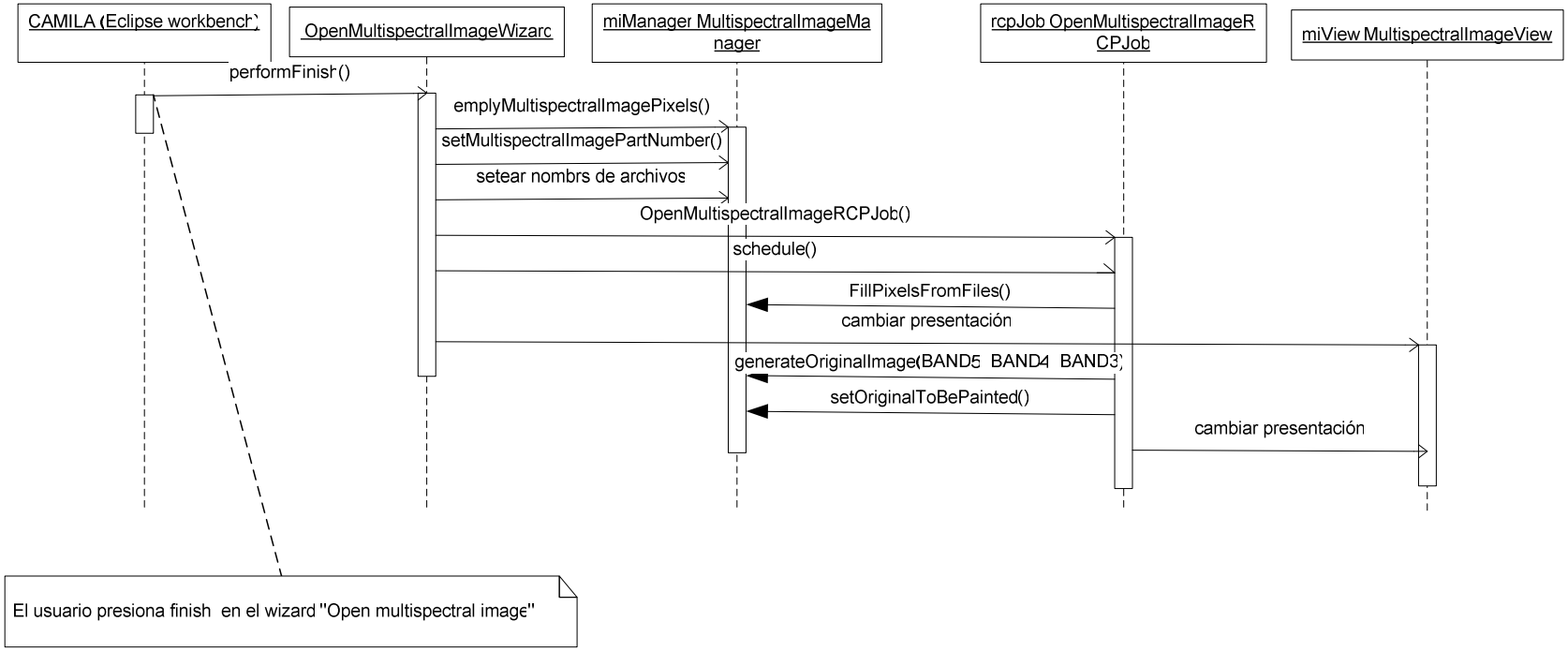

Figura 4.1 Diagrama de secuencia de “Abrir una imagen multiespectral”.

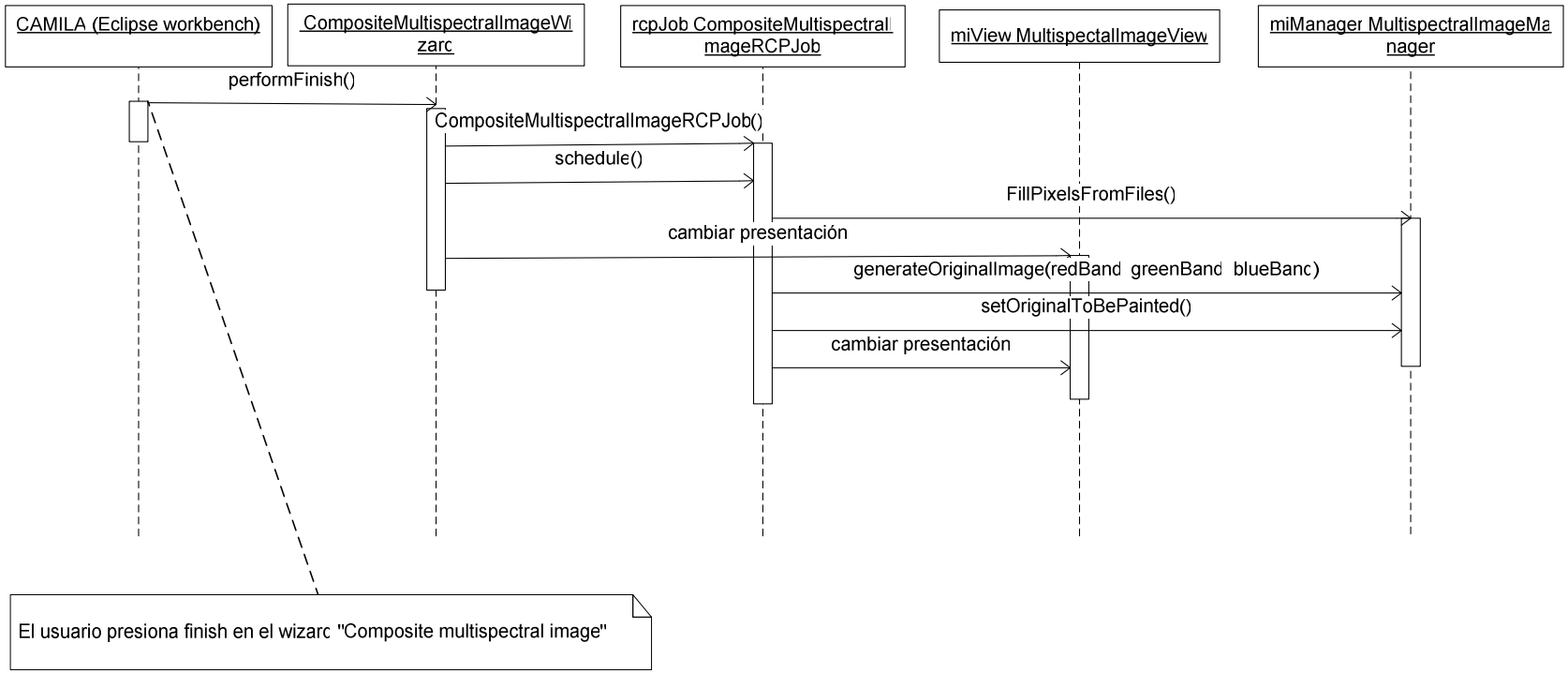

Figura 4.2 Diagrama de secuencia de “Componer una imagen multiespectral”.

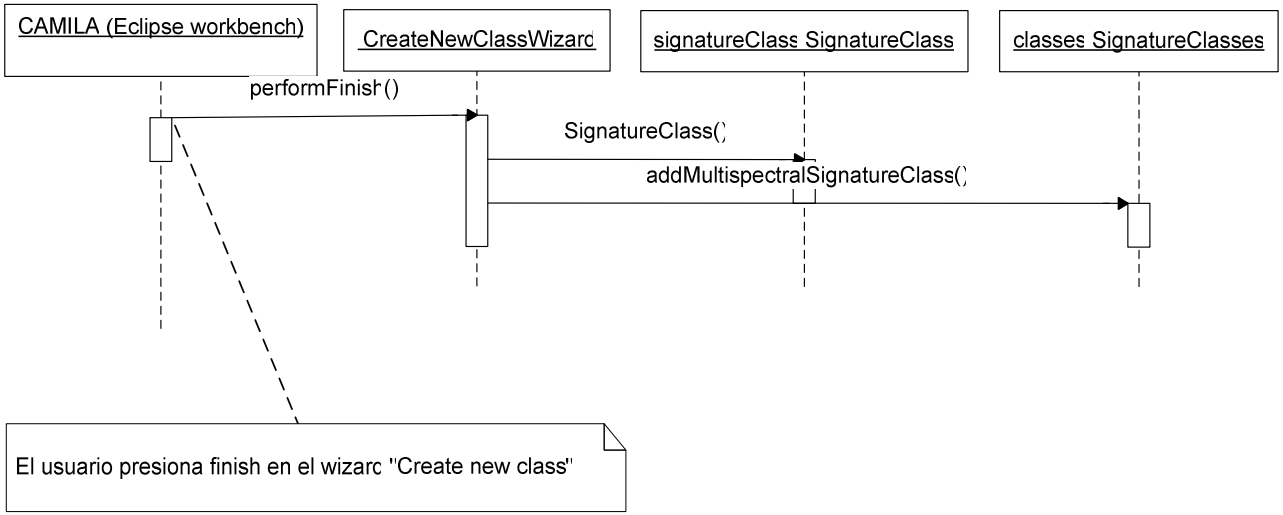

Figura 4.3 Diagrama de secuencia de “Crear clase de píxeles”.

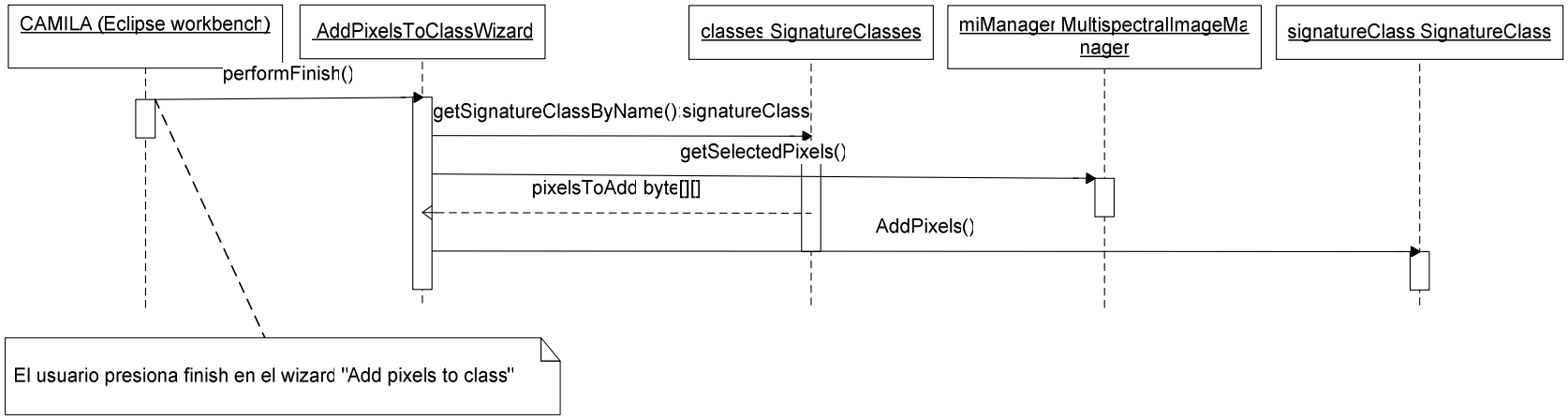

Figura 4.4 Diagrama de secuencia de "Añadir píxeles a una clase".

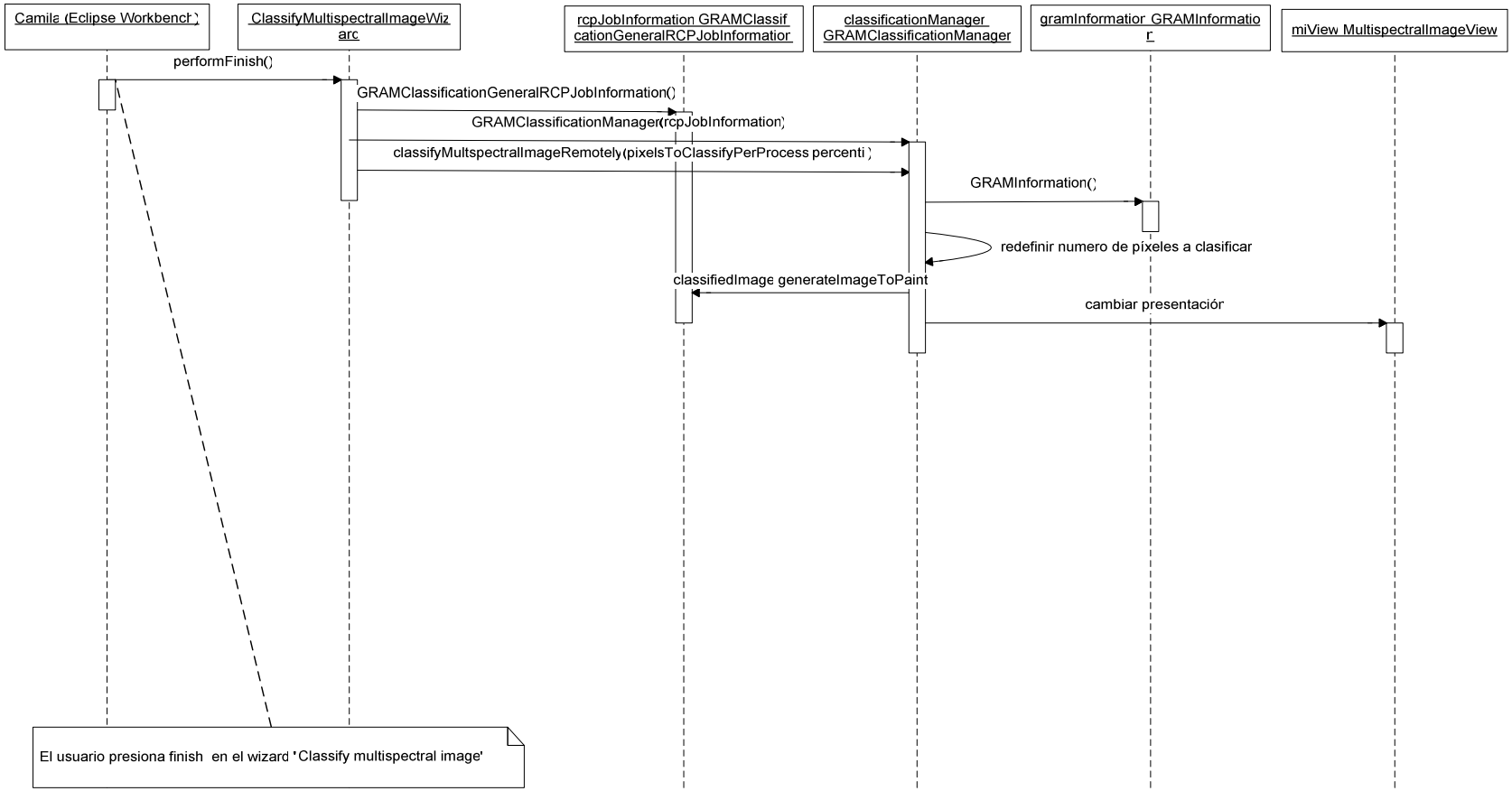

Figura 4.5 Diagrama de secuencia de “Clasificar la imagen multiespectral” parte 1.

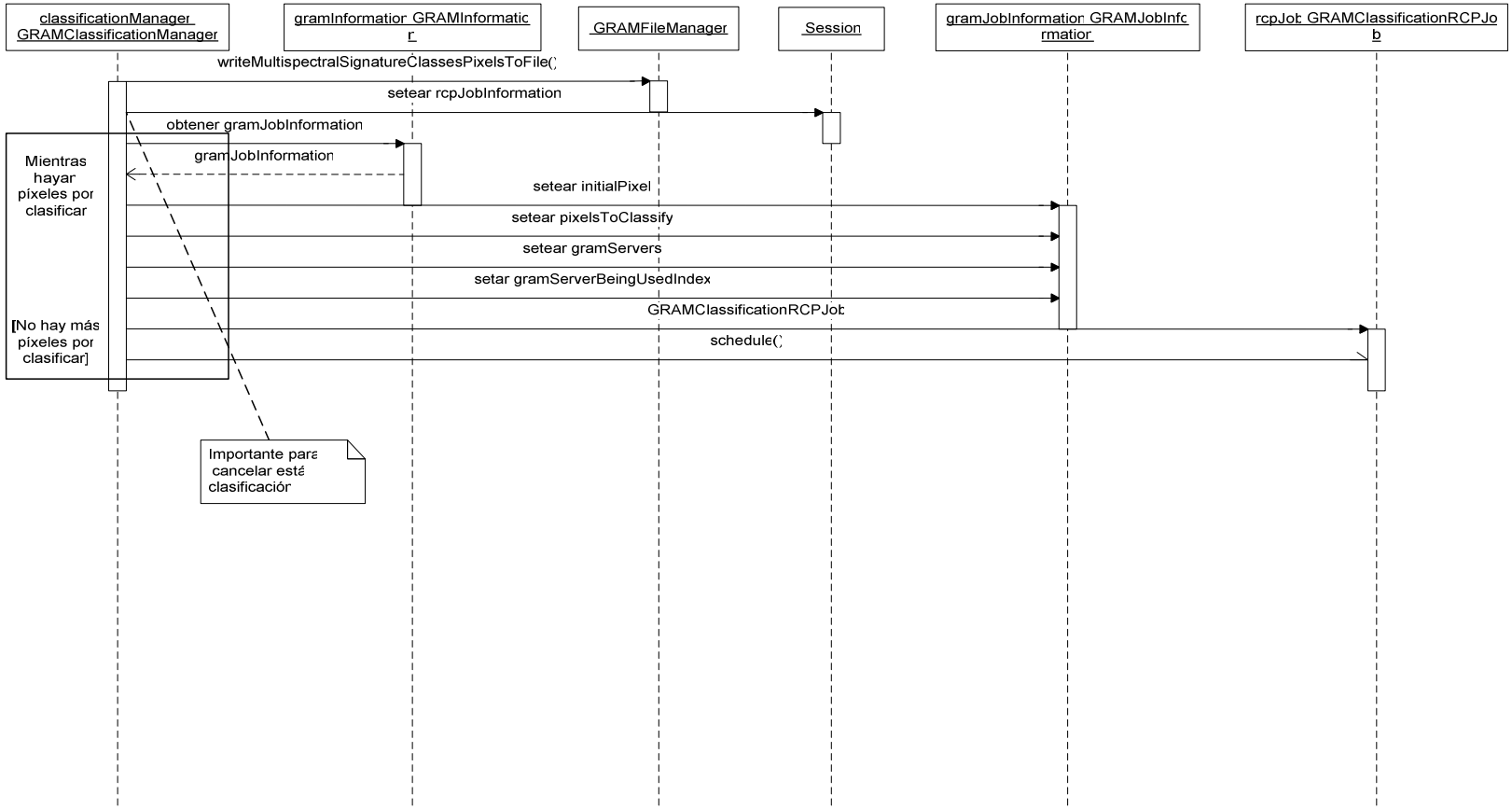

Figura 4.6 Diagrama de secuencia de “Clasificar la imagen multiespectral” parte 2.

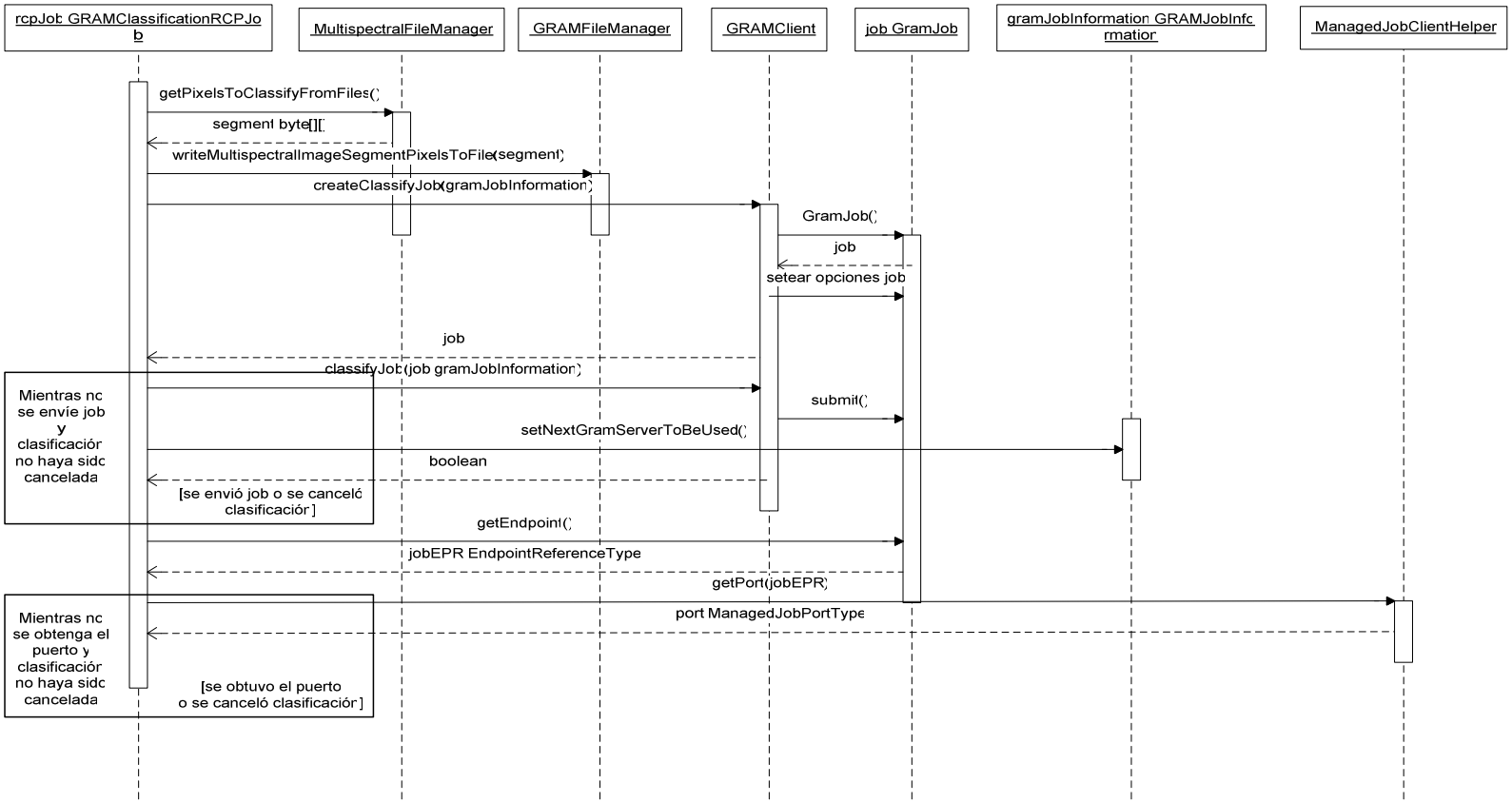

Figura 4.7 Diagrama de secuencia de "Clasificar la imagen multiespectral" parte 3.

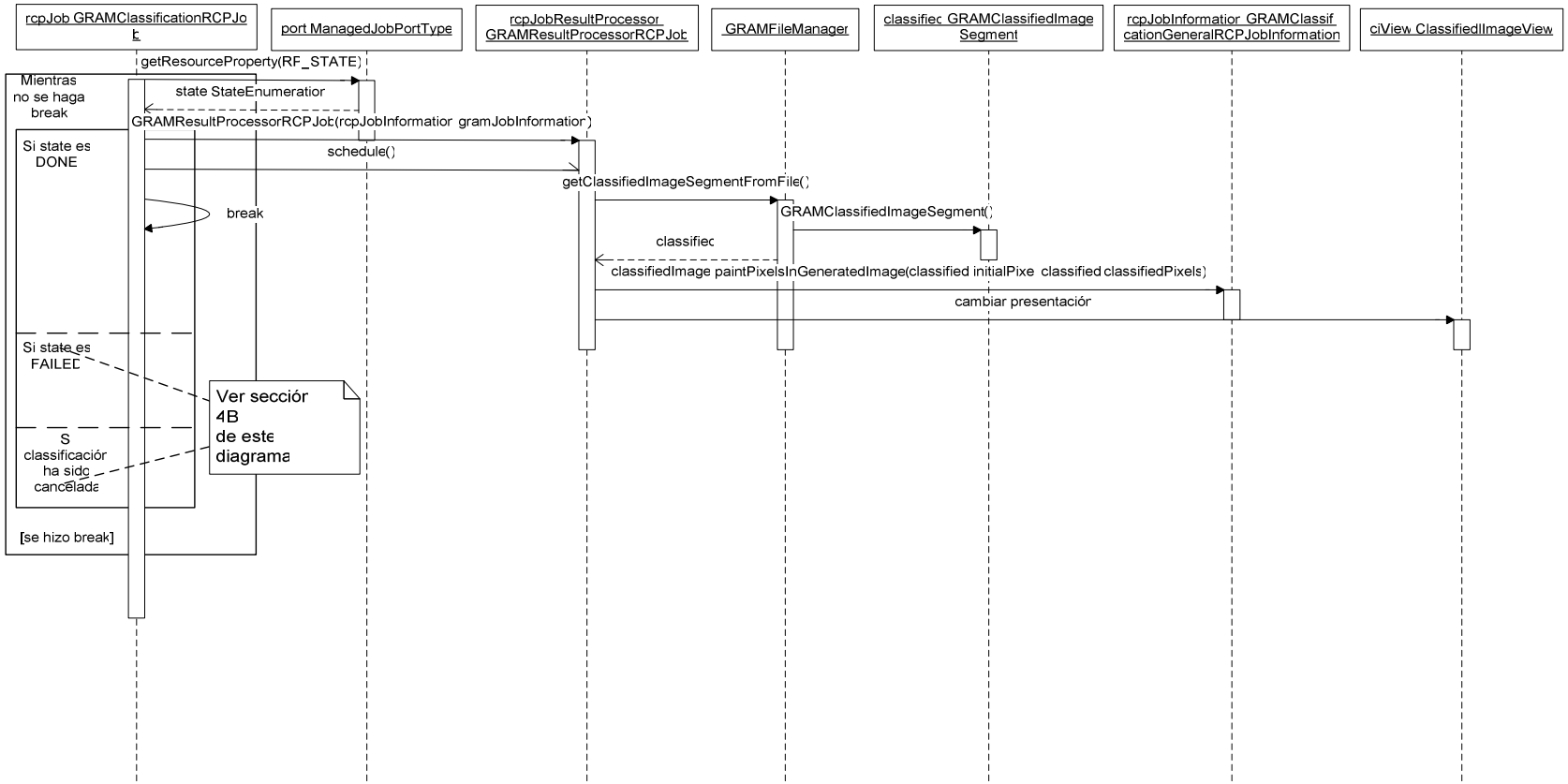

Figura 4.8 Diagrama de secuencia de “Clasificar la imagen multiespectral” parte 4A.

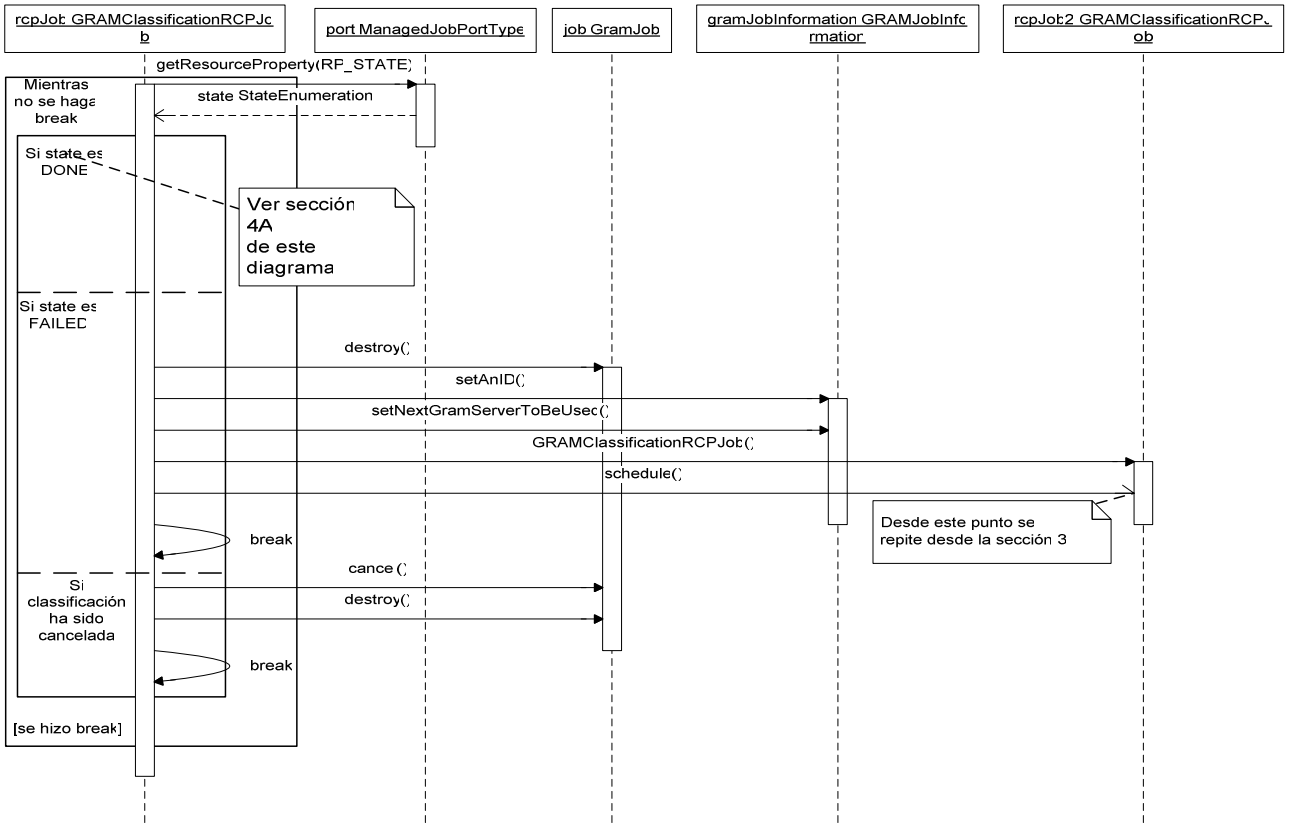

Figura 4.9 Diagrama de secuencia de "Clasificar la imagen multiespectral" parte 4B.

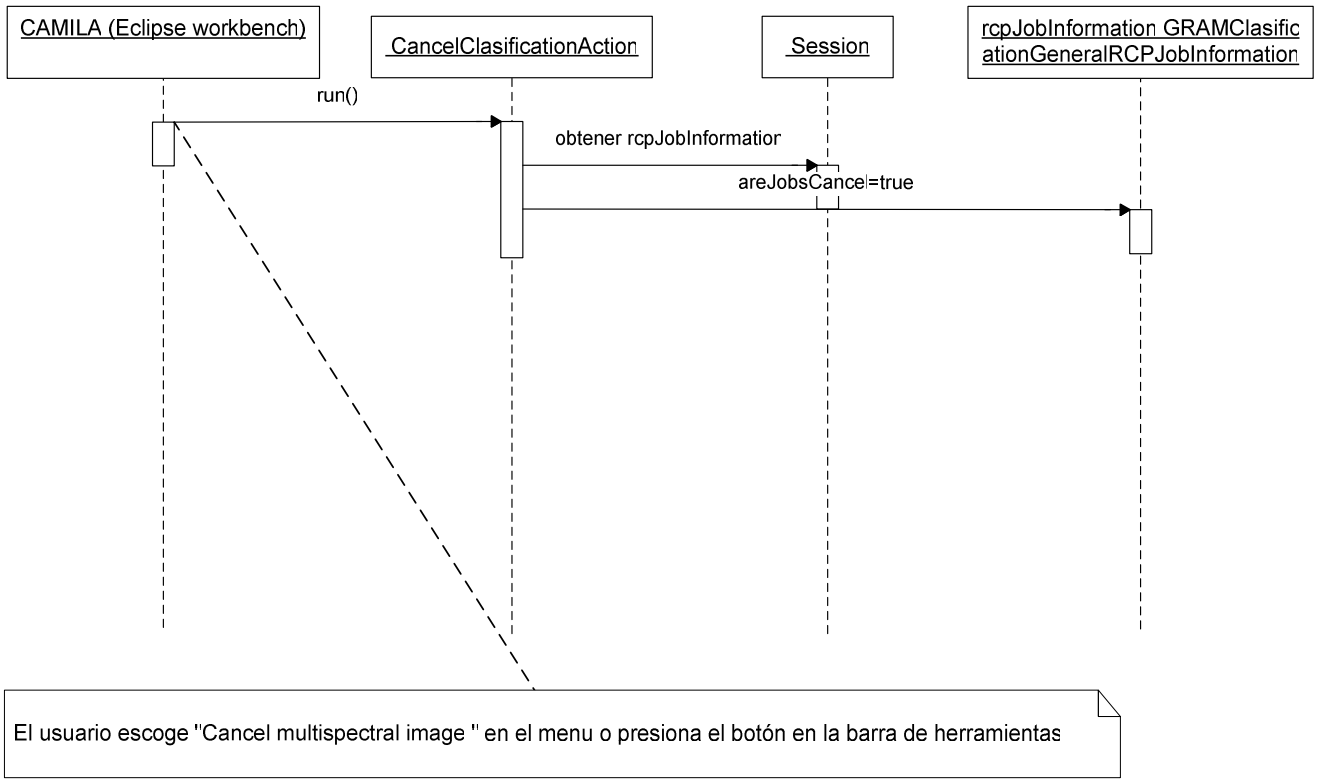

Figura 4.10 Diagrama de secuencia de “Cancelar clasificación”.

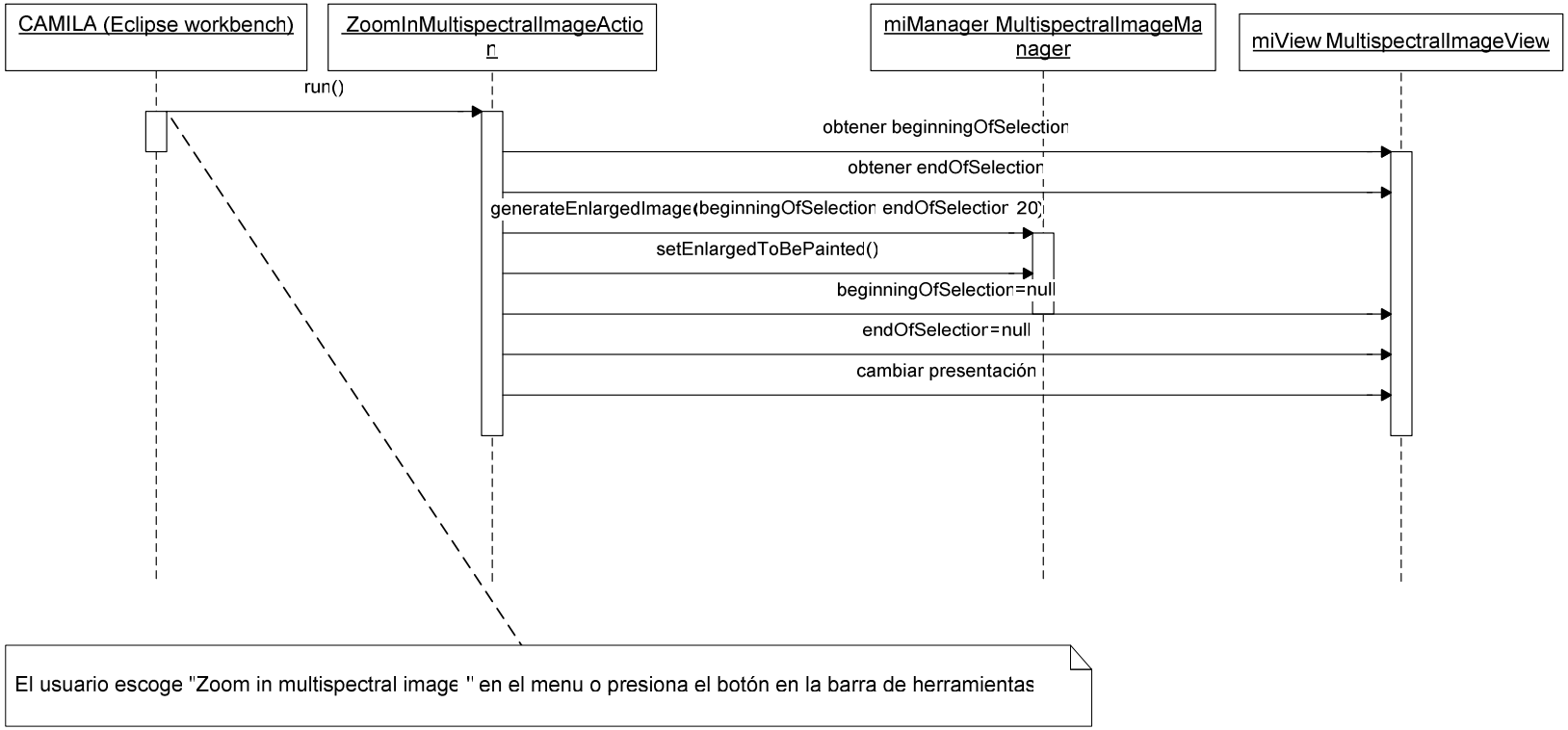

Figura 4.11 Diagrama de secuencia de "Ampliar la imagen multiespectral".

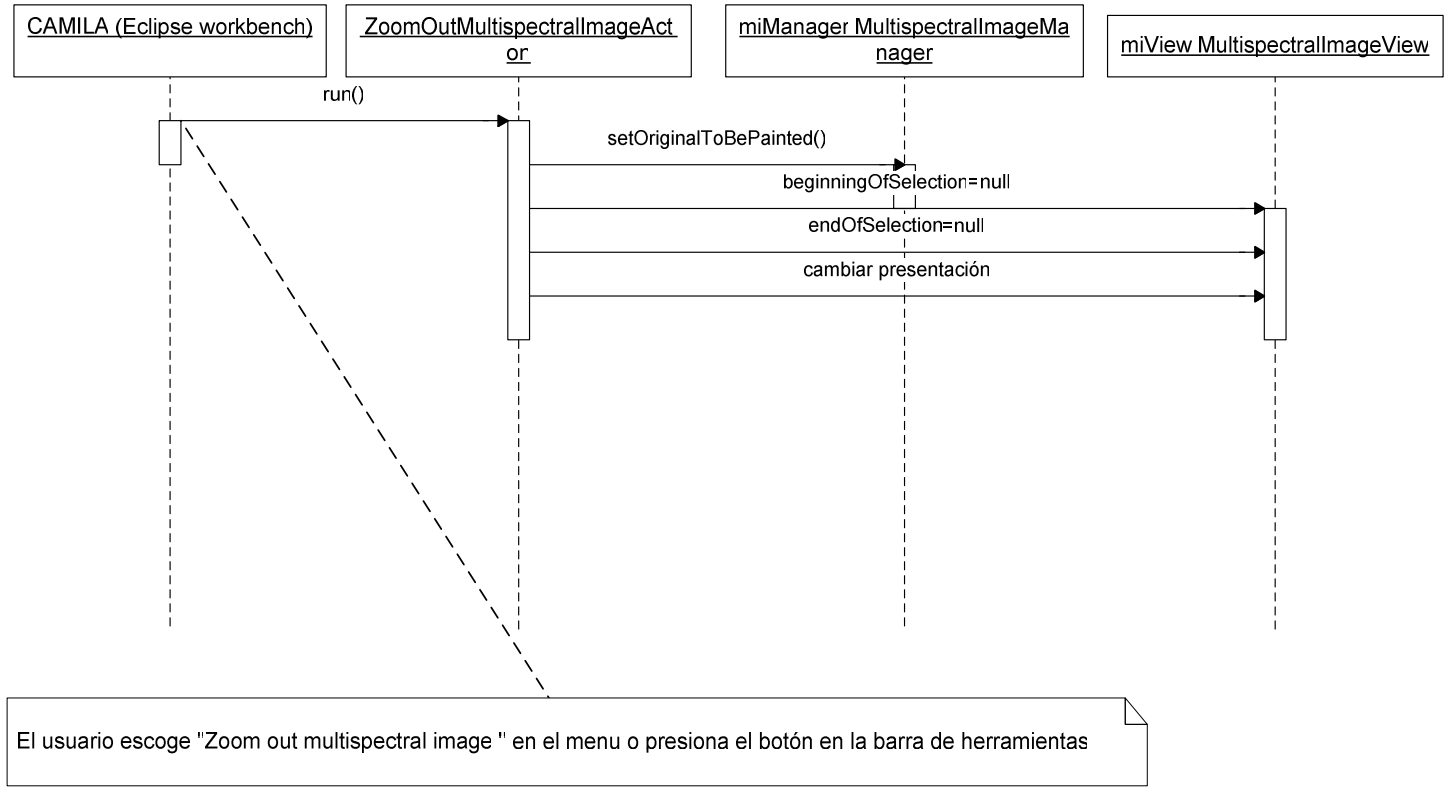

Figura 4.12 Diagrama de secuencia de “Reducir la imagen multiespectral ampliada”.

4.2 INSTALACIÓN DEL AMBIENTE DISTRIBUIDO DE EJECUCIÓN

En este punto se explica cómo está instalado el Grid para soportar el método distribuido de clasificación de imágenes multiespectrales y cómo está implementado el método dentro del ambiente distribuido. El ambiente distribuido lo conforman la aplicación Grid y el Grid.

4.2.1 IMPLEMENTACIÓN DEL GRID

Implementamos un Grid con las capacidades para que soporte el método de clasificación de imágenes multiespectrales usando GT4. Este Grid es un ambiente controlado por una sola administración de recursos, tal que todos los recursos tienen aplicadas las mismas políticas de acceso y seguridad.

4.2.1.1 Componentes GT4 usados

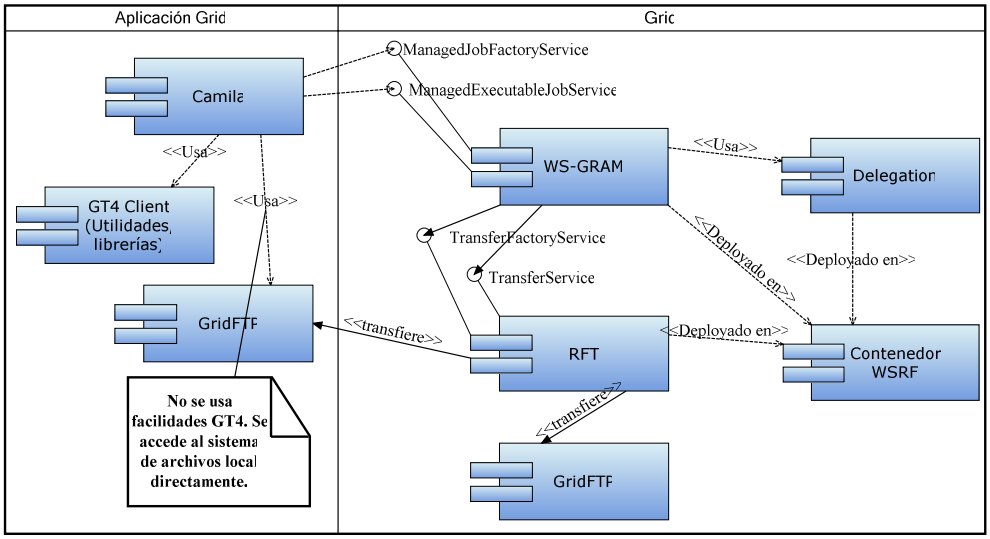

Figura 4.13 Diagrama de componentes del Grid.

En la figura 4.13 se muestran los principales componentes[11] GT4 utilizados. Tenemos dos entidades involucradas: La aplicación Grid y el Grid.

En la Aplicación Grid. El componente CAMILA usa librerías del cliente de WS-GRAM y de WSRF para creación y manejo de jobs; accede al sistema de archivos del servidor GridFTP local a través del sistema de archivos local.

En el Grid. El componente WS-GRAM usa RFT para la transferencia de archivos entre el servidor GridFTP local y el de la entidad CAMILA; usa el componente Delegation para actuar en nombre del usuario para las transferencias de archivos. El contenedor WSRF contiene los servicios correspondientes a los componentes especificados en la figura.

[11] Componente en este contexto es visto como una pieza de software importante. No como un componente en un modelo de componentes.

4.2.1.2 Despliegue de componentes

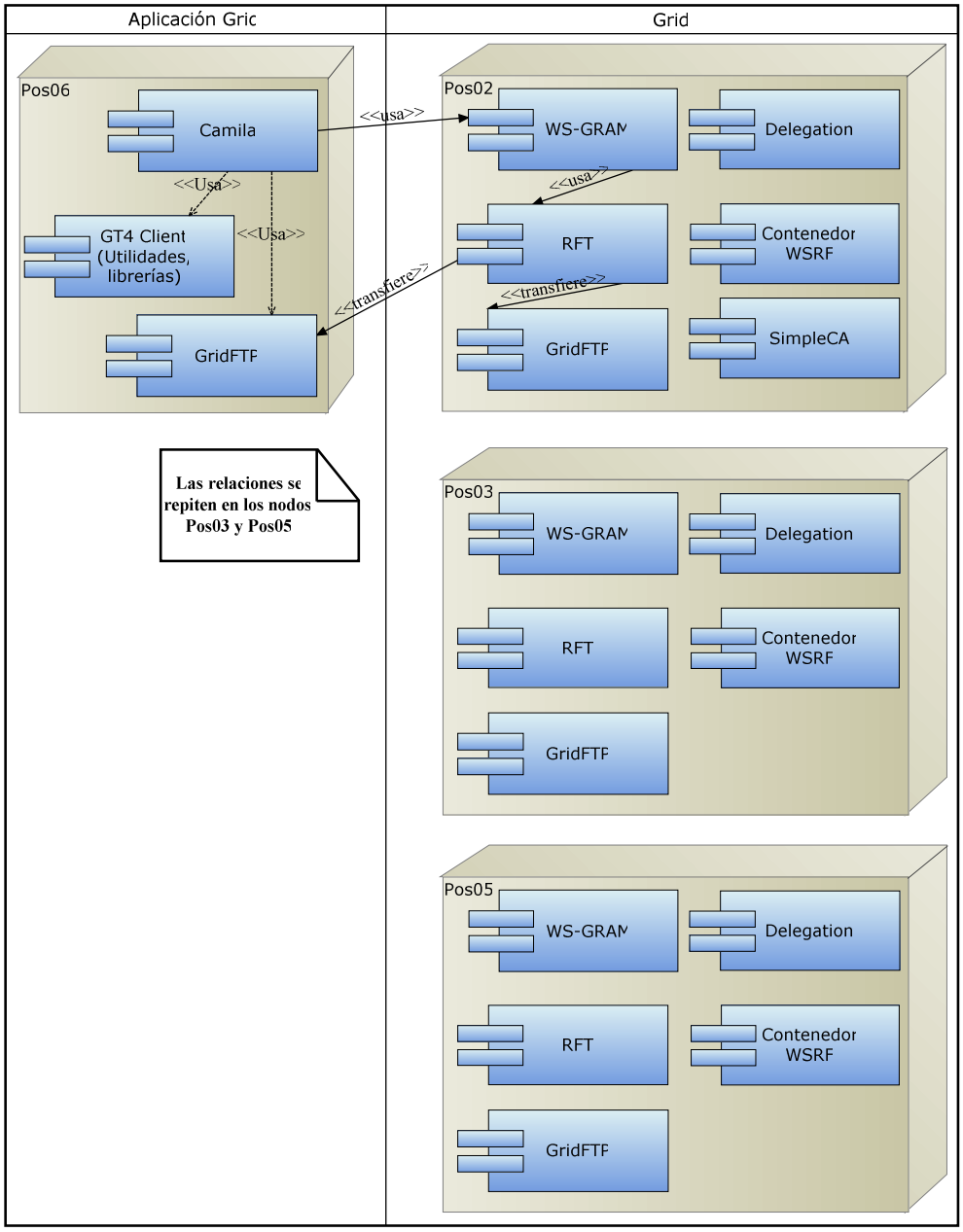

Figura 4.14 Diagrama de despliegue del Grid.

En la figura 4.14 se muestran los nodos con los principales componentes GT4 utilizados. Adicionalmente a los componentes especificados en la figura 4.13, se observa el componente SimpleCA que implementa una autoridad certificadora en el nodo Pos02.

4.2.1.3 Características físicas del Grid

El Grid lo implementamos en el laboratorio Posgrado de los laboratorios del DICC, Departamento de Informática y ciencias de la computación.

El Grid lo conforman tres computadores (Pos02, Pos03, Pos05) en una red LAN Ethernet de 100 Mbps con switch.

Las características de cada computador son:

- Procesador Pentium 4 de 3.2 GHz.
- 1 GB de memoria RAM.
- Sistema operativo Debian Sarge.

4.2.2 IMPLEMENTACIÓN DEL MÉTODO DE CLASIFICACIÓN DE IMÁGENES MULTIESPECTRALES

En la sección 3.2 “Distribución de la computación” describimos el algoritmo de clasificación de imágenes multiespectrales y el escenario de la distribución de la computación. En los diagramas de secuencia de la historia del usuario “Clasificar la imagen multiespectral” de la sección 4.1.4 damos una idea general de su implementación.

En este punto explicamos particularidades de la implementación del método de clasificación de imágenes multiespectrales dentro del ambiente distribuido.

La clasificación de un segmento en un nodo de ejecución es realizada por el método main de la clase MultispectralImageClassifier. Esta clase está empaquetada en el archivo classifier.jar. El método main recibe los argumentos:

- Píxel inicial. Es la posición del primer píxel del segmento en la imagen multiespectral.
- Archivo del segmento. Es la ubicación del archivo que contiene los píxeles a clasificar, en el sistema de archivos local.

- Archivo de las clases. Es la ubicación del archivo que contiene los píxeles de las clases, en el sistema de archivos local.
- Percentil. Usado para calcular la proximidad de un píxel a una clase.
- Archivo del segmento clasificado. Es la ubicación del archivo que contendrá los píxeles clasificados, en el sistema de archivos local.

Como se puede ver, para realizar está clasificación se necesitan los archivos del segmento y de las clases en el nodo de ejecución y el archivo con el resultado de la clasificación necesita ubicarse en el nodo de CAMILA para poder graficar el resultado.

Tomando en cuenta esto, en CAMILA:

Escribimos en un archivo los píxeles de las clases, en el sistema de archivos local.

Por cada job:

- Escribimos en un archivo el segmento (píxeles a clasificar), en el sistema de archivos local.
- Creamos un objeto GramJob, de las librerías cliente de WS-GRAM. Aquí especificamos las transferencias de stagein (archivos de segmento y de clases y el classifier.jar) desde el servidor GridFTP de CAMILA hacia el servidor GridFTP del nodo del servidor WS-GRAM y las transferencias de stageout (archivo con resultado de la clasificación) en dirección inversa. También especificamos el programa ejecutable, “java”, y los argumentos del programa ejecutable, entre otras cosas.
- Enviamos el job a un servidor WS-GRAM.
- Periódicamente consultamos el estado del job con librerías del cliente de WSRF:
    - Si el estado es DONE, quiere decir que ya se realizó la fase de stageout y ya tenemos el segmento clasificado en el sistema de archivos local. Entonces graficamos el resultado.

- Si el estado es FAILED, reenviamos el job a otro servidor WS-GRAM. El servidor al que se envía es simplemente el siguiente en una lista, no se usan criterios de cargas, ya que normalmente un servido WS-GRAM no es un nodo de ejecución.

4.3 PRUEBAS DE FUNCIONAMIENTO

En este punto se muestran los resultados de diferentes pruebas del funcionamiento del método distribuido de clasificación de imágenes multiespectrales.

4.3.1 DESCRIPCIÓN DE LAS PRUEBAS

Las pruebas consisten en diferentes casos de clasificaciones de una imagen multiespectral. En cada caso se harán clasificaciones locales y en el Grid. La diferencia entre los casos será la cantidad de procesamiento, definida por el número de píxeles de las clases.

Para cada caso se tomarán los tiempos de respuesta.

Las pruebas no tienen como objetivo encontrar un patrón o función en base a los diferentes casos, ni justificar que la clasificación en el Grid es mejor que la local. Su propósito es dar una idea de la distribución de la carga en el Grid implementado.

4.3.2 ESTIMACIÓN DE LA COMPUTACIÓN A DISTRIBUIRSE

La cantidad de computación a distribuirse es igual al número de operaciones involucradas en la clasificación que serán distribuidas. Este valor depende del tamaño de la imagen y del número píxeles de las clases.

La cantidad de computación a distribuirse para una imagen de 23'461.888 píxeles (correspondiente a la imagen proporcionada por CLIRSEN) y un conjunto de seis clases con 50 píxeles cada una, podría estimarse en 49''269'964.800 operaciones (sin tomar en cuenta todos los pasos del algoritmo sino solamente las restas entre valores de bandas). Justificado de la siguiente manera:

7 restas entre valores de índices de reflexión de píxeles.

Multiplicado por 23,461.888 numero de píxeles de la imagen

Multiplicado por 300 número de píxeles de las clases

Da como resultado 49''269'964.800 operaciones.

4.3.3 FACTORES INFLUYENTES EN LAS PRUEBAS

En las pruebas realizadas hay factores que influyen considerablemente en el tiempo de respuesta de las pruebas.

Factores constantes:

- Los nodos de ejecución son los mismos servidores WS-GRAM.
- Tenemos 3 servidores/nodos de ejecución.
- Los nodos se encuentran en comunicación cercana. LAN Ethernet de 100 Mbps con switch.
- Los nodos son dedicados. Los nodos no realizan otra actividad importante más que las involucradas en la clasificación.
- Procesador Pentium 4 de 3.2 GHz.
- 1 GB de memoria RAM.

Factores que hemos decidido sean constantes.

- Imagen a usar. A pesar de que tenemos dos imágenes, hemos decidido usar solo una, la más grande, debido que requiere más

cómputo su clasificación. Esta es una de las variables que influye en el número de tareas de clasificación.

- Tamaño del segmento a clasificar, número de píxeles a clasificar, por tarea. A pesar de que la aplicación permite setear este valor, hemos decidido que su valor sea 782000. Este valor fue obtenido empíricamente, por permitir un alto número de jobs sin congestionar la red, es decir sin generar otro cuello de botella. Esta es la otra variable que influye en el número de tareas de clasificación.
- El número de tareas de clasificación será de 30 dados la imagen y el número de píxeles a clasificar por segmento.

Factores variables:

- Número total de píxeles de las clases. Como se puede ver en el punto 4.3.2 “Estimación de la computación a distribuirse”, dada una imagen multiespectral, la cantidad de procesamiento es mayormente afectada por el número total de píxeles de las clases.

4.3.4 RESULTADOS DE LAS PRUEBAS

Caso	Píxeles de clases	Tipo de clasificación	Tiempo de respuesta
1	60	Local	03’ 41”
			03’ 48”
		Grid	04’ 27”
			04‘ 22“
2	255	Local	15’ 44 “
			15‘ 31“
		Grid	07’ 17”
			07‘ 33”
3	372	Local	24’ 07‘’
			23’ 50”
		Grid	10’ 14”

			10‘ 21”
4	612	Local	37’ 44”
			37’ 55”
		Grid	16‘ 12”
			17’ 38”

Tabla 4.14 Resultado de pruebas.

Observaciones importantes de las pruebas:

En la segunda prueba tipo Grid del caso 4 falló el job 0 en el nodo pos02 por lo que se reenvió al nodo pos03. Al minuto 16 ya habían terminado todos los jobs en los otros nodos. Esto hizo que el tiempo de respuesta de la clasificación de toda la imagen sea mucho mayor al caso en que no hay error.

El tipo de clasificación local se la hace con hilos de ejecución en la misma máquina donde corre CAMILA y se tiene un lock para el acceso a los archivos de las imágenes.

En todos los casos tipo Grid a pesar de ya haberse realizado la clasificación y la ubicación del resultado en la maquina de CAMILA (stageout), la consulta del estado fallaba muchas veces por lo que el job no aparecía como DONE en la aplicación y por lo tanto no se graficaba el resultado tan pronto como el job terminaba. Esto se debe a la congestión en la red y a que las consultas se hacen usando el protocolo UDP.

Solo en el caso 1 la clasificación local toma menos tiempo que la clasificación tipo Grid.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

En esta sección se presentan las conclusiones y recomendaciones obtenidas durante la realización de la tesis.

5.1 CONCLUSIONES

CAMILA puede ser migrada a otros Grids GT4 con pocos cambios. CAMILA usa el componente WS-GRAM para distribuir la carga, este es el componente estándar de GT4 para manejo de ejecución. CAMILA tiene parametrizado en la interfaz de usuario el número de píxeles a clasificar por job, de esto depende el número de jobs. CAMILA tiene parametrizadas opciones de jobs WS-GRAM en su código fuente.

En CAMILA fue importante tomar en cuenta la memoria usada por la aplicación. La imagen más grande proporcionada por CLIRSEN, correspondiente a un cuarto de una imagen total, tiene un tamaño 156 MB. CAMILA está pensada para un computador con 1 GB de memoria. Manejar la información de las imágenes de manera estructurada en memoria ocupa más espacio que la imagen original.

En un Grid real no es común plantear un esquema de reenvío de jobs. En un Grid real el job se envía a un calendarizador de un conjunto de nodos. El calendarizador es el responsable del reenvío de un job a otro nodo si ocurre una falla o cancelación. Solo si el calendarizador no pudiera usarse, se necesitaría un esquema de reenvío de jobs. En nuestro Grid tenemos un esquema sencillo de reenvío de jobs debido a que el mismo servidor WS-GRAM es el nodo de ejecución.

Para que el resultado de una clasificación sea fiable es importante definir el

conjunto de píxeles de una clase con píxeles característicos de la clase. A pesar de que CAMILA garantiza que un mismo píxel no sea añadido a diferentes clases, píxeles muy parecidos pueden ser añadidos a diferentes clases, esto podría afectar el cálculo de la pertenencia de un píxel de la imagen multiespectral a una clase. Si cada clase tiene solo píxeles característicos de su clase, no es importante la diferencia entre el número de píxeles entre clases, debido a que la distribución de distancias entre los píxeles de la clase y el píxel a clasificar será similar.

El criterio del percentil para el cálculo de la distancia entre un píxel y una clase es más fiable que sacar un píxel promedio de la clase y calcular la distancia entre los dos píxeles. El sacar el píxel promedio de los píxeles de una clase ahorra cómputo pero ese valor está afectado por todos los píxeles de la clase, incluyendo los píxeles no tan característicos. El criterio del percentil ignora las distancias entre el píxel a clasificar y los píxeles de las clases que están fuera del percentil.

GT4 no podría proveer un calendarizador de tareas. El principio detrás de la computación Grid es la interacción entre organizaciones. GT4 da facilidades para la integración de las organizaciones. Ya que cada organización es responsable por administrar sus recursos, WS-GRAM funciona como una capa sobre calendarizadores sin responsabilizarse por la administración de recursos.

En todo Grid real son importantes los requerimientos de seguridad. Conceptualmente un Grid real está formado por instituciones autónomas que al hacer uso del Grid exponen datos y comunicaciones que son solo de su interés en un ambiente interinstitucional. En el mismo Grid, se tienen diferentes requerimientos de seguridad para diferentes casos, ya que en el mismo Grid los miembros de las organizaciones tienen diferentes intereses en diferentes casos (usos del Grid); por ejemplo en un Grid entre universidades puede haber usuarios o aplicaciones que puedan acceder solo a ciertos recursos del Grid.

Los parámetros de un job WS-GRAM no son genéricos para cualquier Grid. Hay parámetros que dependen de la comunicación entre el nodo que envía el job y el servidor WS-GRAM. Hay parámetros que dependen del servidor WS-GRAM y del calendarizador del que haga uso.

Para el desarrollo de aplicaciones Grid computacionales usando WS-GRAM es necesario entender GT4 en general. Casi todo GT4 está implementado como servicios de WSRF (incluyendo WS-GRAM) y se podría requerir implementar servicios WSRF propios. WS-GRAM hace uso de los mecanismos de seguridad implementados por el Grid. WS-GRAM usa RFT y GridFTP para las transferencias de archivos. WS-GRAM usa Delegation para la ubicación y transferencia de archivos y para la ejecución de tareas. Se podría tener un servicio de Index de un conjunto de servidores WS-GRAM.

El tiempo de desarrollo de CAMILA fue afectado mayormente por carecer de un conjunto de clases y sus píxeles. El método de clasificación distribuida de imágenes multiespectrales necesita un conjunto de clases y sus píxeles. CAMILA necesitó definir un conjunto de clases y sus píxeles; esto implicó que las imágenes multiespectrales deban ser manejadas gráficamente (visualizadas, compuestas, ampliadas, seleccionadas una región). De haber tenido un conjunto de clases y sus píxeles CAMILA habría usado una interfaz de línea de comandos.

El tamaño del segmento de imagen a clasificar en una tarea influye en el tiempo de respuesta. El tiempo de computación no depende del número de segmentos, pero a más segmentos, más procesos, más accesos a disco y más transferencias en la red; estos pueden generar cuellos de botella.

5.2 RECOMENDACIONES

GT4 es complejo y su documentación es escasa. Debido a esto, para usar GT4 se requiere tener experiencia en programación, habilidades de lectura

de código y experiencia en sistemas tipo Unix.

En base a nuestra experiencia, recomendamos la metodología XP para proyectos de exploración de tecnologías (en nuestro caso RCP y manejo de imágenes multiespectrales). XP está pensado en el manejo de requerimientos cambiantes. En un proyecto de exploración de tecnologías los requerimientos no son claramente definidos, poco entendidos y cambiantes.

CAMILA puede servir como una aplicación de ejemplo del uso de RCP, WS-GRAM y WSRF. En CAMILA se hace uso de varias capacidades RCP (wizards, actions, jobs, products, etc), SWT (GridLayout, images). En CAMILA se usan librerías del cliente de WS-GRAM para envío y cancelación de jobs y de WSRF para consulta de estado de recursos.

En nuestra búsqueda de un problema que requiera mucho cómputo y sea altamente paralelizable, es decir una aplicación apta para computación distribuida, encontramos varios. Los siguientes problemas pueden ser usados en trabajos que tengan que ver con distribución de carga: pruebas exhaustivas de código, minado de datos, renderizado de imágenes.

Para que esta aplicación utilice de manera óptima recursos computacionales, debería ser usada en un Grid que use WS-GRAM como componente de distribución de carga junto con un calendarizador de recursos que maneje al menos tantos recursos como procesos de clasificación.

GLOSARIO

A

Acción (RCP). Es una forma de contribuir comportamiento al Workbench. Con las acciones se puede contribuir ítems a los menús y a la barra de herramientas. Comúnmente una acción activa un wizard.

API. (Application Programming Interface). Interface de Programación de Aplicaciones. Es una interface que un sistema o librería de programa provee para soportar peticiones de servicios por parte de un programa.

C

Clase (imágenes multiespectrales). Es un conjunto de píxeles que cumplen cierto criterio asociativo. En la aplicación el criterio es dado por parte del usuario al momento de crear clases y añadir píxeles a las mismas. **Clasificar una imagen multiespectral.** Es calcular la correspondencia de cada píxel de una imagen multiespectral a una clase.

E

Eclipse Workbench (rcp). Eclipse Workbench es la parte básica de RCP. Proporciona el soporte para la infraestructura de plugins y funciona como un motor sobre el cual se añaden funcionalidades.

EPR (GT4). (End Point Reference). Dentro de WS-Addressing, es una construcción diseñada para referenciar un servicio Web con su recurso asociado. Siendo el recurso la manera estándar de guardar información de estado dentro de WSRF. Un EPR es el Servicio Web + el recurso.

Extension (RCP). Es una implementación de un punto de extensión. En CAMILA todos los puntos de extensión son de la plataforma Eclipse.

F

Framework. Es una estructura de soporte sobre la cual se desarrolla software. Puede incluir programas, librerías de código, software para ayudar

en el desarrollo, etc.

G

GNOME. Ambiente de escritorio para sistemas tipo Unix, es el oficial del proyecto GNU.

GNU/LINUX. Sistema operativo GNU con Linux como kernel.

Grid map file. Es un archivo que contiene entradas que mapean los subjects de un certificado a nombres de cuentas/usuarios locales.

GTK+2. Es un widget toolkit para el sistema X Window para crear interfaces de usuario gráficas. Es usado por GNOME.

H

Historia del usuario (XP). Es un requerimiento de software formulado en una o dos frases en lenguaje natural.

I

IDE. (Integrated development environment). Ambiente de desarrollo integrado. Es un tipo de software que asiste a programadores en el desarrollo de software.

Imagen multiespectral. Imagen ópticamente adquirida en más de un espectro o intervalo de longitud de onda. Una imagen multiespectral es un conjunto de datos de índices de reflexión en diferentes frecuencias (bandas) de un conjunto de áreas determinadas. Un área determinada se denomina píxel.

J

Job (GT4). Es una tarea computacional que puede ejecutar operaciones de entrada y salida mientras corre, lo cual afecta el estado del recurso computacional y su sistema de archivos.

Job scheduler. Un calendarizador de tareas es una aplicación de software que se encargan de ejecuciones en el background no asistidas, comunmente

conocidas como procesamiento batch.

P

Plataforma. Describe una especie de framework, en hardware o en software, que permite que corra software.

Poco acoplado. (Loosely coupled). En cuanto a componentes, son poco acoplados cuando es fácil componer componentes en otros componentes e intercambiar componentes con similares. Se logra al hacer a las interfaces del componente con asumpciones mínimas haciendo que si se hace un cambio a un componente no se deba cambiar sus dependientes.

Producto (RCP). Es la manera en que el Ambiente de Desarrollo de Plugins (PDE) maneja a la aplicación como un producto desde el punto de vista de marketing. No tiene que ver con funcionalidad, más bien con presentación. Un producto es implementado extendiendo el punto de extensión org.eclipse.core.runtime.products y usando un archivo de configuración.

R

RCP. Rich Client Platform. Es la plataforma sobre la que está construido el IDE Eclipse. Proporciona un ambiente sobre el cual construir aplicaciones de escritorio usando el lenguaje Java y SWT, entre otras tecnologías.

Refactoring (XP). Proceso de mejora del diseño del código y de su implementación llevado a cabo en una iteración.

S

Servicio Web. Es un sistema de software que soporta interacción entre agentes de software. Tiene una interface descrita en WSDL. Otros sistemas interactúan con el Servicio Web usando mensajes SOAP típicamente usando HTTP con XML. No es ejecutable, depende de otras tecnologías para su ambiente de ejecución.

Stagein (GT4). Dentro de WS-GRAM es la etapa en la que un job enviado realiza las transferencias de archivos necesarios para la ejecución del

mismo.

Stageout (GT4). Dentro de WS-GRAM es la etapa en la que un job enviado realiza las transferencias de archivos después de haberse ejecutado.

SWT. (Standard Widget Toolkit). Es el widget toolkit usado por el framework RCP. Accede a las librerías GUI nativas del sistema operativo usando JNI.

T

Tarea de ingeniería (XP). Es una tarea relativa a la implementación de una historia del usuario que hay que llevar a cabo en una iteración.

Throughput. Cantidad de datos por unidad de tiempo. Tasa a la cuál un computador o red envía o recibe datos.

V

View (RCP). Una vista es una sección de despliegue de información.

W

Wizard (RCP). Es una ventana que maneja una secuencia de pasos para ejecutar una tarea.

X

X.509. Es un estándar para infraestructura de clave pública. Especifica, entre otras cosas, estándares para certificados y el algoritmo de validación del certificado.

XP. (eXtreme Programming). Es una metodología de ingeniería de software. Se diferencia de metodologías tradicionales principalmente en que da mayor valor a la adaptabilidad que a la predictibilidad. Su principal objetivo es reducir el costo de los cambios. El cambio en los requerimientos es natural, deseable y realista. No se trata de definir todos los requerimientos al inicio del proyecto.

En un proyecto XP se inicia con la solución más simple y se hace refactoring a soluciones mejores. El diseño y la codificación se enfocan en necesidades

actuales y el diseño del sistema crece y mejora a medida que se avanza en el proyecto. No se tiene un diseño total antes de empezar; si se desconoce las tecnologías, hacer un diseño de todo no es posible ya que no se sabe ni las posibles ni las mejores soluciones para los problemas. El diseño evolutivo funciona debido a que no es costoso hacer cambios.

BIBLIOGRAFÍA

TESIS:

[1] BRITO, Carmen; GAVILANES, Verónica. Introducción a las tecnologías grid computing y data grid. Código: 02010731101. Julio 2004.

LIBROS:

[2] BERMAN, Fran; FOX, Geoffrey; HEY, Tony. Grid Computing, Making the Global Infrastructure a Reality. Wiley. ISBN: 0-470-85319-0. 2003.

[3] FOSTER, Ian; KESSELMAN, Carl. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann. ISBN: 1-55860-475-8. 1999.

[4] NEWCOMER, Eric; LOMOW, Greg. Understanding SOA with Web Service. Addison-Wesley. ISBN: 0-321-18086-0. 2005.

WHITE PAPERS:

[5] FOSTER, Ian; KESSELMAN Carl; TUECKE Steven. The Anatomy of the Grid. 2001.

[6] FOSTER, Ian; KESSELMAN Carl; TUECKE Steven; NICK Jeffrey. The Physiology of the Grid. Junio 2002.

[7] FOSTER, Ian. What is the Grid? A Three Point Checklist. Julio 2002.

[8] FOSTER, Ian. Globus Toolkit Version 4: Software for Service-Oriented Systems. 2005.

[9] INOSTROZA, Pablo. Una Introducción a los Grid Services. 2005

[10] BARCENA; BECERRA; FERNANDEZ. Entorno de supercomputación Grid para aplicaciones de altas demandas computacionales. Enero 2004

[11] Foster; Kishimoto; Savva; Berry; Djaoui; Grimshaw; Horn; Maciel; Siebenlist; Subramaniam; Treadwell; Reich.The Open Grid Services Architecture, Version 1.0. Enero 2005.

[12] CZAJKOWSKI; FERGUSON; FOSTER; FREY; GRAHAM; SEDUKHIN; SNELLING; TUECKE; VAMBENEPE. The WS-Resource Framework. Mayo 2004.

[13] GRAHAM; NIBLETT; CHAPPELL; LEWIS; NAGARATNAM; PARIKH; PATIL; SAMDARSHI; STEVE TUECKE; VAMBENEPE; WEIHL. Web Services Notification (WS-Notification). Enero 2004.

[14] BOX; CHRISTENSEN; CURBERA; FERGUSON; FREY; HADLEY; KALER; LANGWORTHY; LEYMANN; LOVERING; LUCCO; MILLET; MUKHI; NOTTINGHAM; ORCHARD; SHEWCHUK; SINDAMBIWE; STOREY; WEERAWARANA; WINKLER. Web Services Addressing (WSAddressing). Agosto 2004.

GUÍAS:

[15] SOTOMAYOR Borja. The Globus Toolkit 4 Programmer's tutorial. 2005.

[16] FOSTER, Ian. A Globus Primer. Or, Everything You Wanted to Know about Globus, but Were Afraid To Ask. Describing Globus Toolkit Version 4. Mayo 2005.

[17] Globus Alliance community. GT4 Admin Guide. Febrero 2005.

WEB:

[18] Globus Alliance. Documentación referente a Grid Computing y Globus Toolkit 4. http://www.globus.org.

[19] http://www-128.ibm.com/developerworks/grid/library/gr-overview/.

[20] http://gridcafe.web.cern.ch/gridcafe/whatisgrid/whatis.html.

[21] http://www.microsoft.com/latam/net/basics/glossary.asp

[22] http://www.globus.org/ogsa/

[23] http://gridcenter.or.kr/GridInfra/02.php

[24] http://www.extremeprogramming.org

[25] http://www.gisdevelopment.net/glossary

[26] http://fsweb.olin.edu/~mchang/research/nasa/

[27] http://fcf.unse.edu.ar/pdf/lpr/p9.PDF

ANEXOS

ANEXO No. 1 MANUAL DE USUARIO DE CAMILA

Figura A-1 Splash screen de CAMILA.

1. ABRIR UNA IMAGEN MULTIESPECTRAL.

Se escoge “Open multispectral image” en el menú “File” o en el icono de la barra de herramientas.

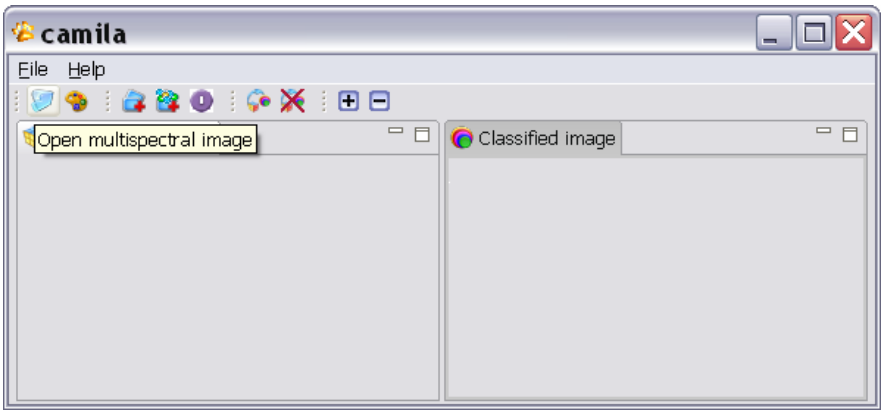

Figura A-2 Escoge “Open multispectral image”.

En el wizard: se ubica el archivo “header.txt” de la imagen multiespectral a abrir, se selecciona la parte de la imagen a cargar y los archivos correspondientes a cada una de las bandas.

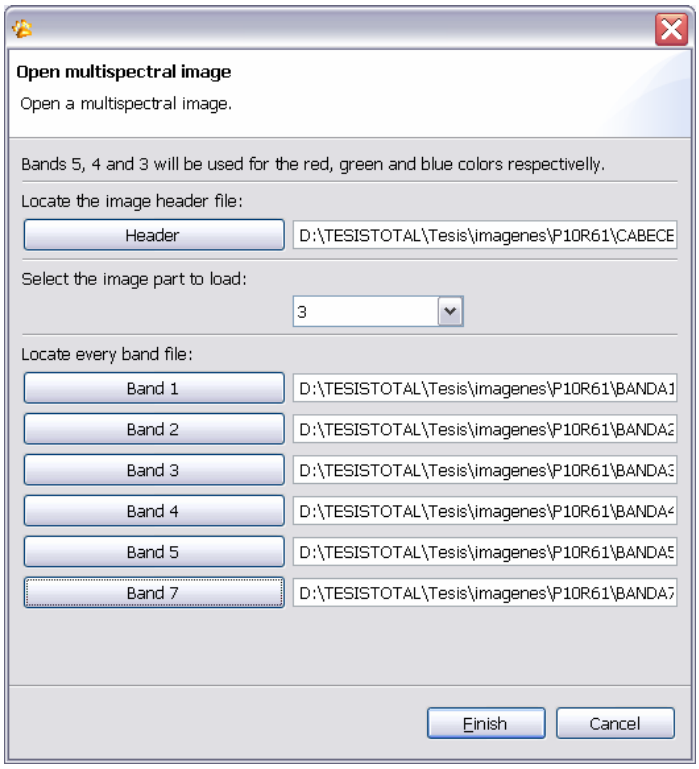

Figura A-3 Wizard "Open multispectral image".

Se presiona "Finish". CAMILA abrirá la imagen en el background, desplegando un mensaje en la sección "Multispectral image" que indica que la imagen se está abriendo.

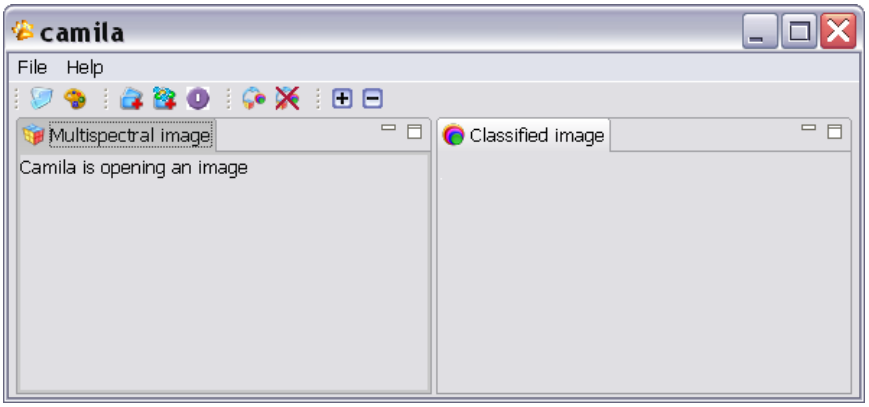

Figura A-4 Abriendo "Open multispectral image".

C

Una vez que la imagen ha sido abierta se despliega un mensaje informándo del hecho.

Figura A-5 Mensaje abierto "Open multispectral image".

La imagen abierta presenta en su parte superior información relevante acerca de la imagen y su graficación.

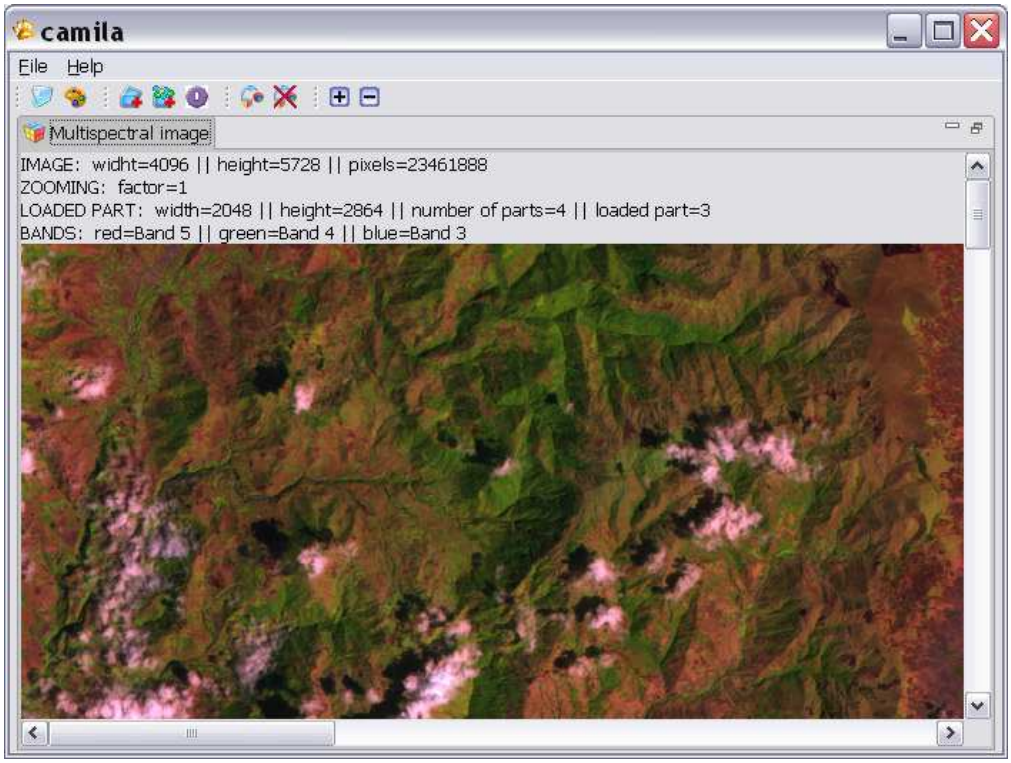

Figura A-6 Imagen abierta "Open multispectral image".

2. COMPONER LA IMAGEN MULTIESPECTRAL.

Se escoge "Composite multispectral image" en el menú "File" o en el icono de la barra de herramientas.

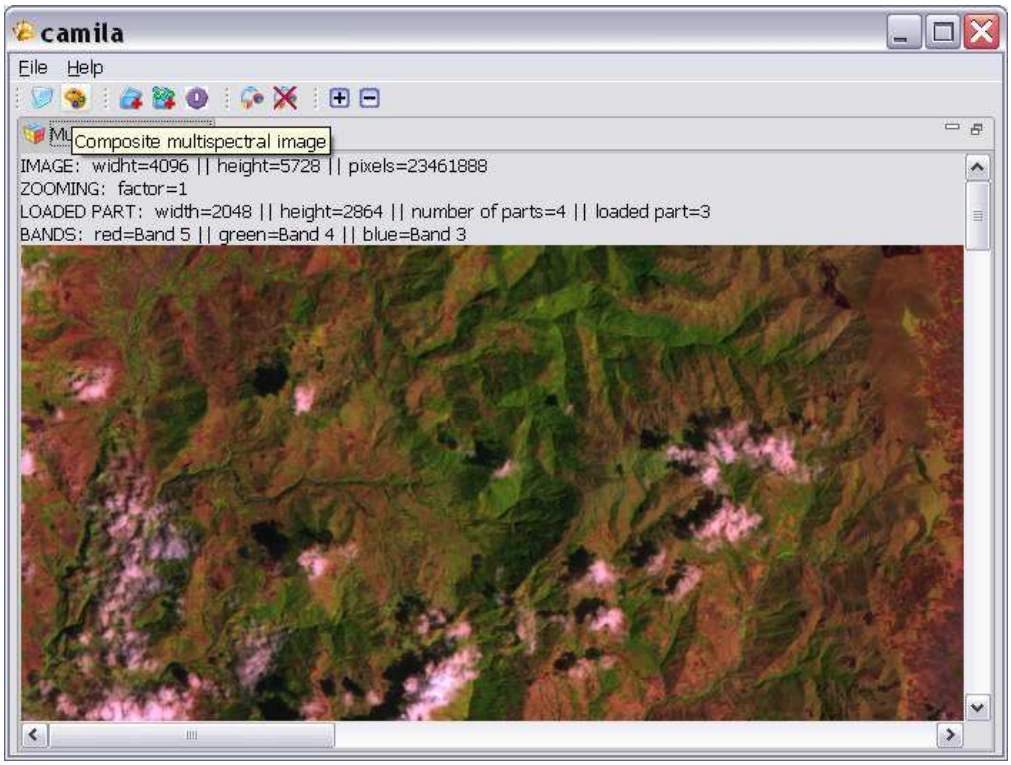

Figura A-7 Escoge "Composite multispectral image".

En el wizard: se selecciona las bandas correspondientes a los colores rojo, verde y azul.

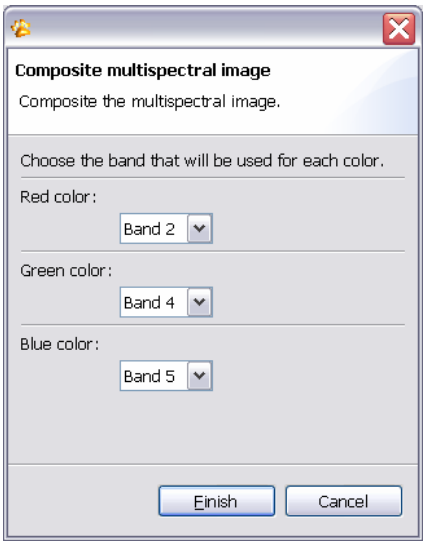

Figura A-8 Wizard "Composite multispectral image".

Se presiona “Finish”. CAMILA compondrá la imagen en el background, desplegando un mensaje en la sección “Multispectral image” que indica que la imagen se está componiendo.

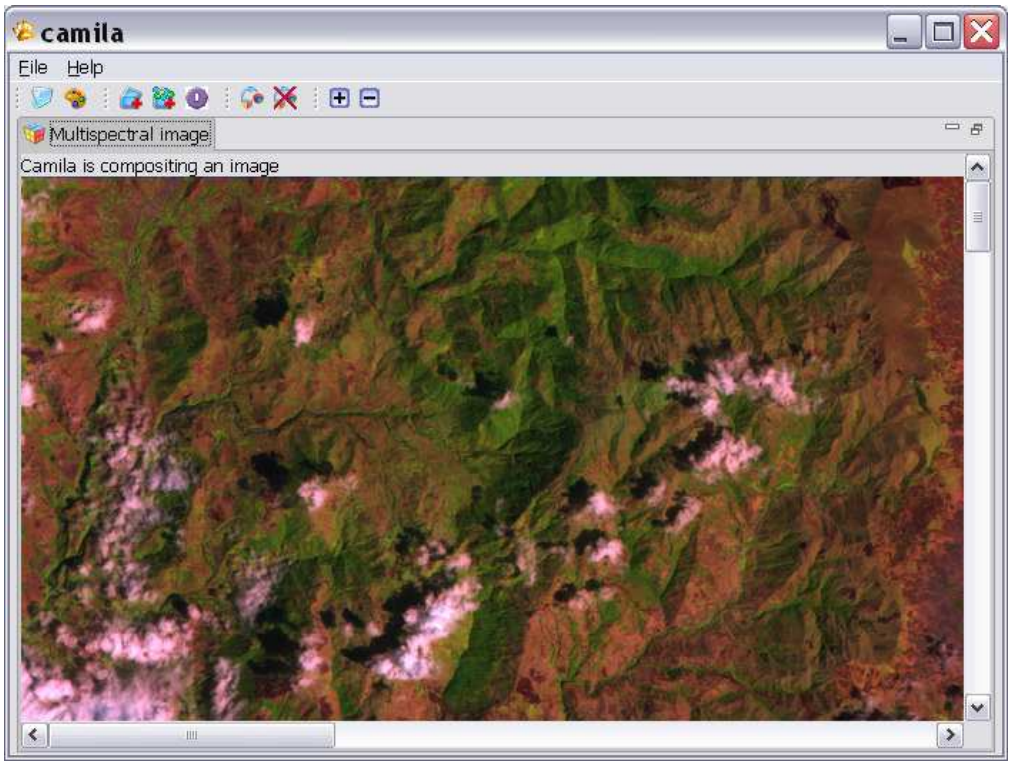

Figura A-9 Componiendo “Composite multispectral image”.

Una vez que la imagen ha sido compuesta se despliega un mensaje informando del hecho.

Figura A-10 Mensaje compuesta “Composite multispectral image”.

La imagen compuesta presenta en su parte superior información relevante acerca de la imagen y su graficación.

F

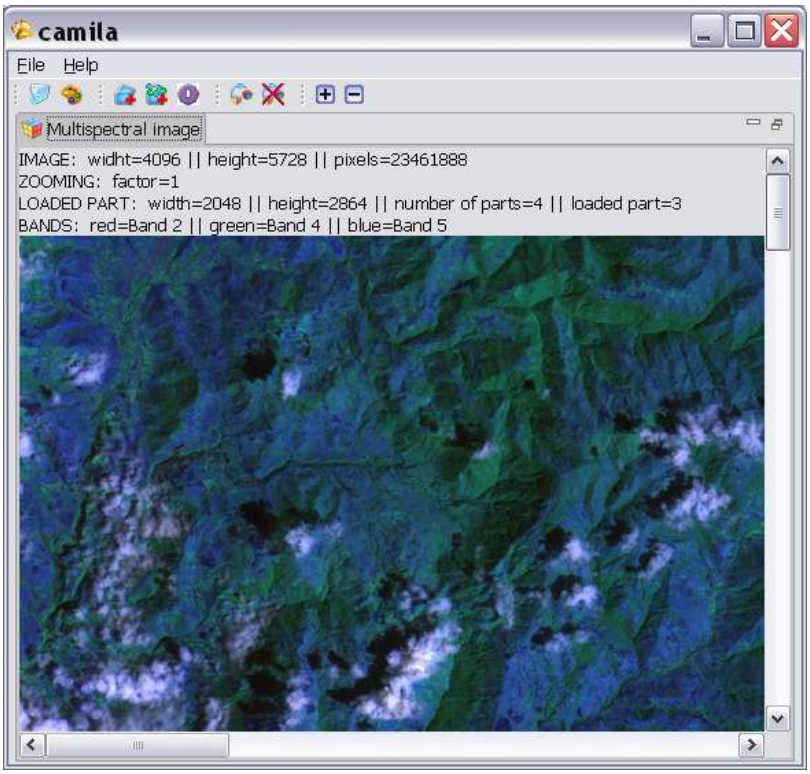

Figura A-11 Imagen compuesta "Composite multispectral image".

3. CREAR UNA CLASE DE PÍXELES.

Se escoge "Create new class" en el menú "File" o en el icono de la barra de herramientas.

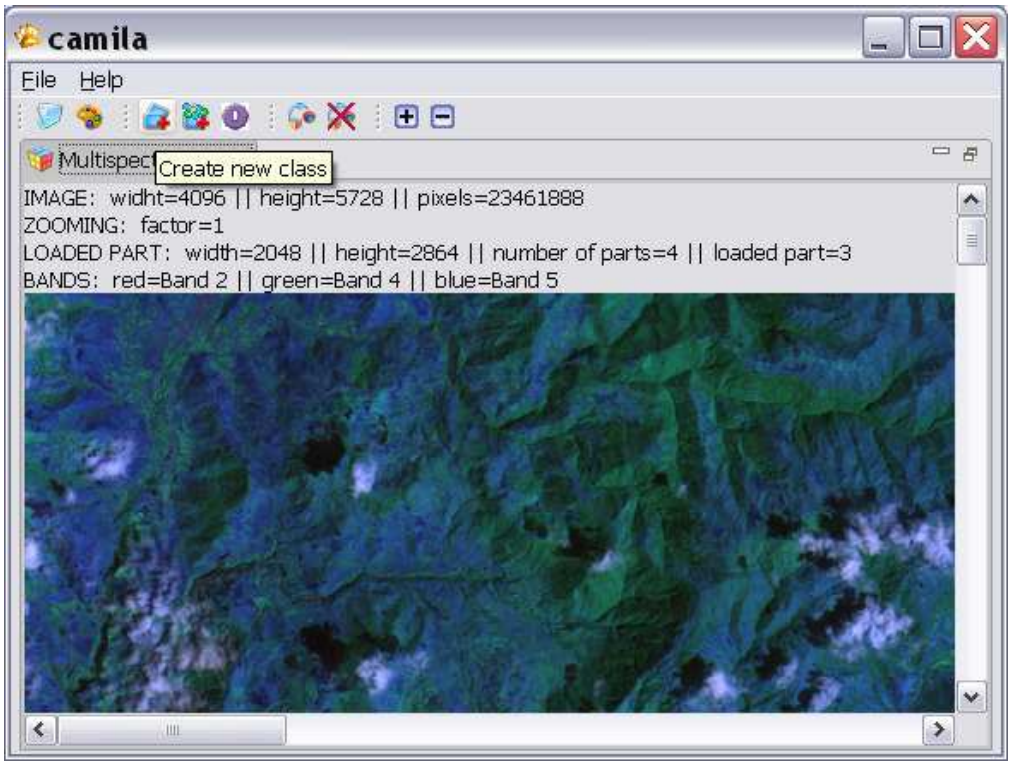

Figura A-12 Escoge "Create new class".

En el wizard: se escribe el nombre de la clase y se escoge el color con el que se graficarán los píxeles de que correspondan a esa clase en la imagen clasificada.

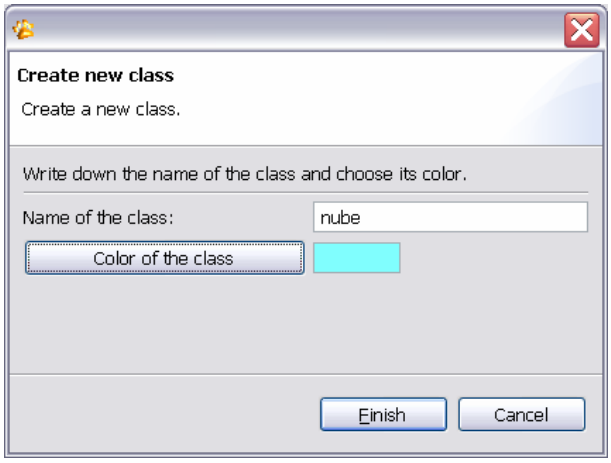

Figura A-13 Wizard "Create new class".

Se presiona "Finish". CAMILA crea la clase con un conjunto de píxeles vacío.

4. AÑADIR PÍXELES A UNA CLASE.

Se selecciona un área de la imagen multiespectral graficada (puede ser de la imagen tamaño original o ampliada) y se escoge “Add pixels to class” en el menú “File” o en el icono de la barra de herramientas.

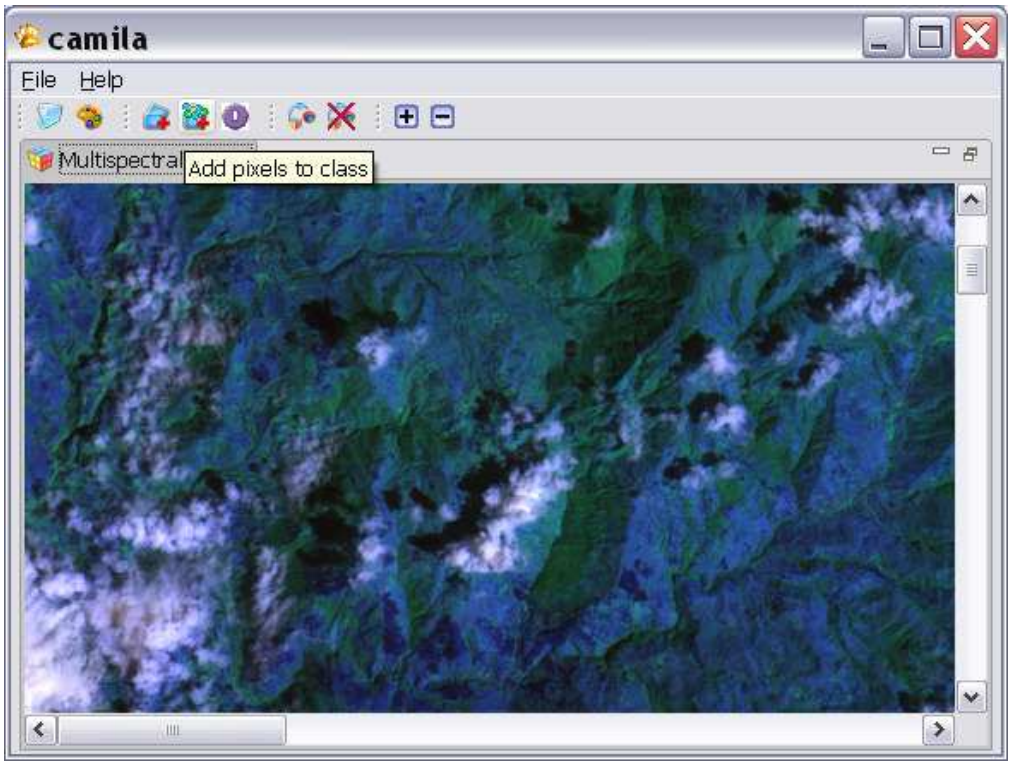

Figura A-14 Escoge “Add pixels to class”.

En el wizard: se escoge el nombre de la clase a la que queremos añadir los píxeles seleccionados.

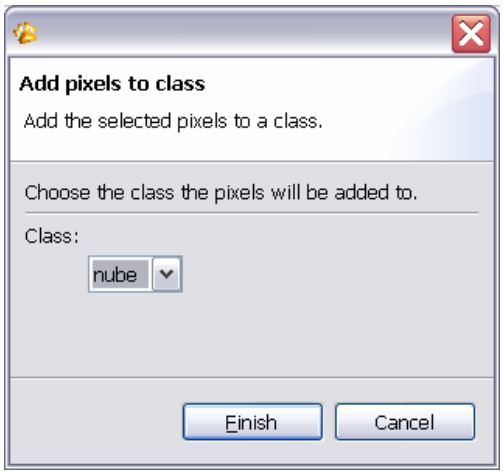

Figura A-15 Wizard "Add pixels to class".

Se presiona "Finish". CAMILA añade los píxeles seleccionados a la clase escogida.

5. MOSTRAR INFORMACIÓN DE LAS CLASES.

Se escoge "Show classes information" en el menú "File" o en el icono de la barra de herramientas.

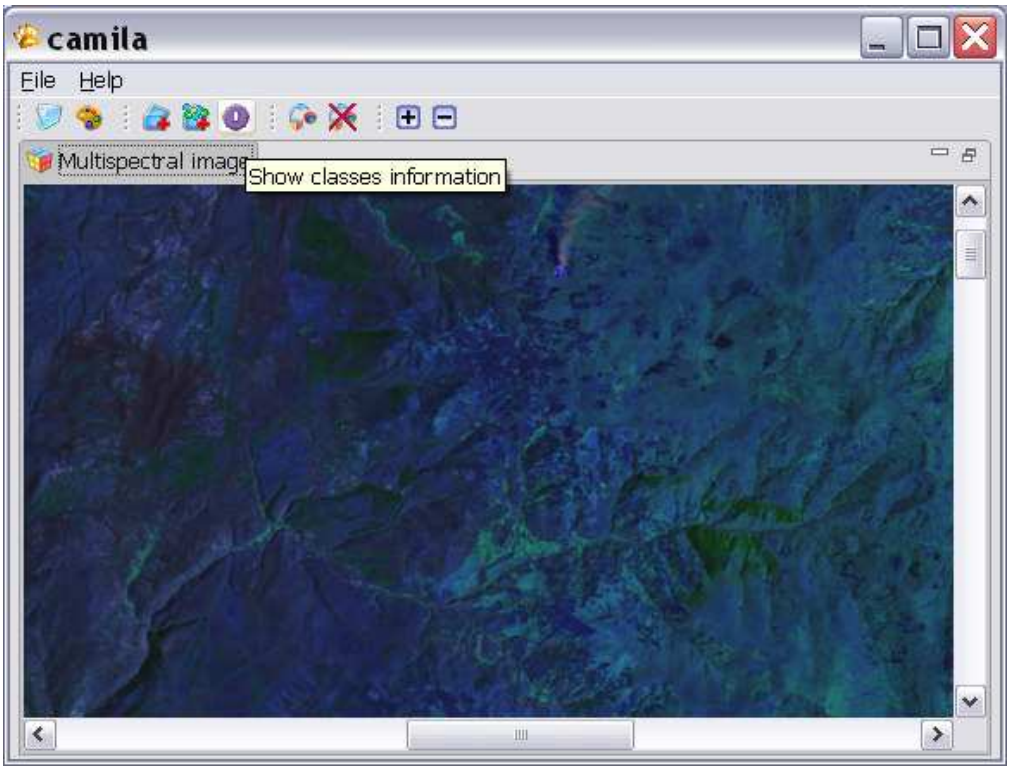

Figura A-16 Escoge "Show classes information".

Se muestra información acerca de las clases creadas.

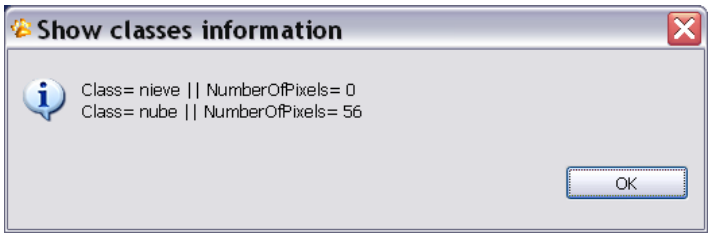

Figura A-17 Mensaje información "Show classes information".

6. AMPLIAR LA IMAGEN MULTIESPECTRAL.

Se selecciona un área de la imagen multiespectral en tamaño original y se escoge "Zoom in multispectral image" en el menú "File" o en el icono de la barra de herramientas.

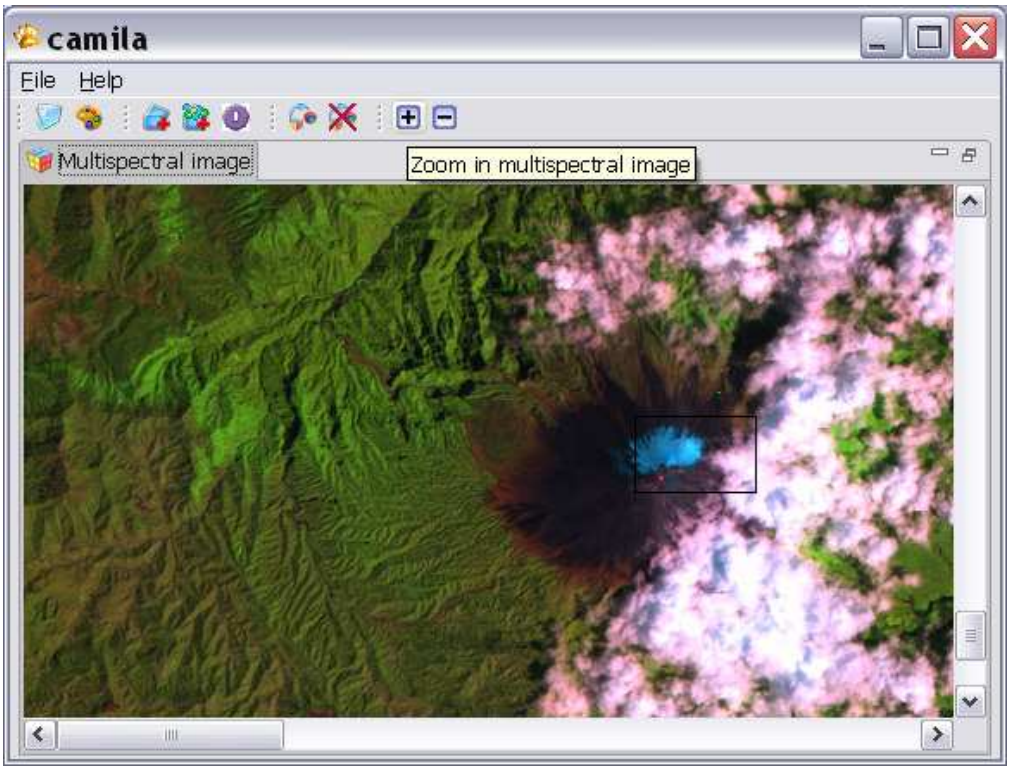

Figura A-18 Escoge "Zoom in multispectral Image".

Se grafica la imagen ampliada y en su parte superior se presenta información relevante acerca de la imagen y su graficación.

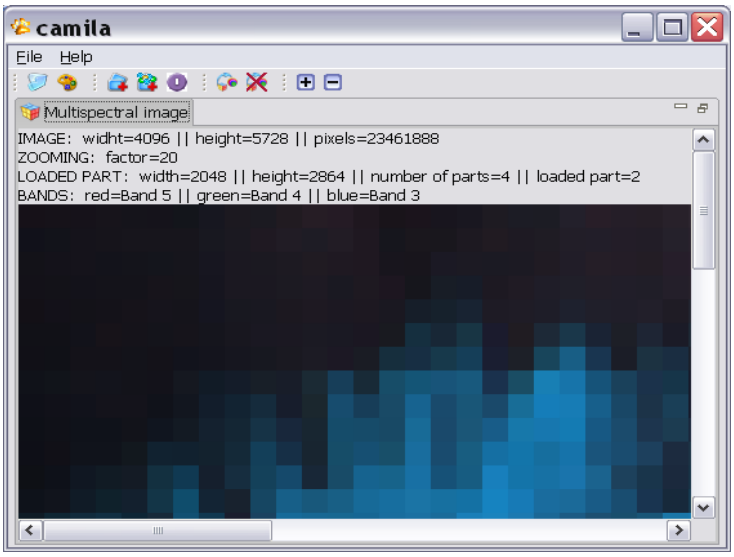

Figura A-19 Imagen ampliada "Zom in multispetral image".

7. REDUCIR LA IMAGEN MULTIESPECTRAL AMPLIADA.

Estando la imagen ampliada, se escoge “Zoom out multispectral image” en el menú “File” o en el icono de la barra de herramientas.

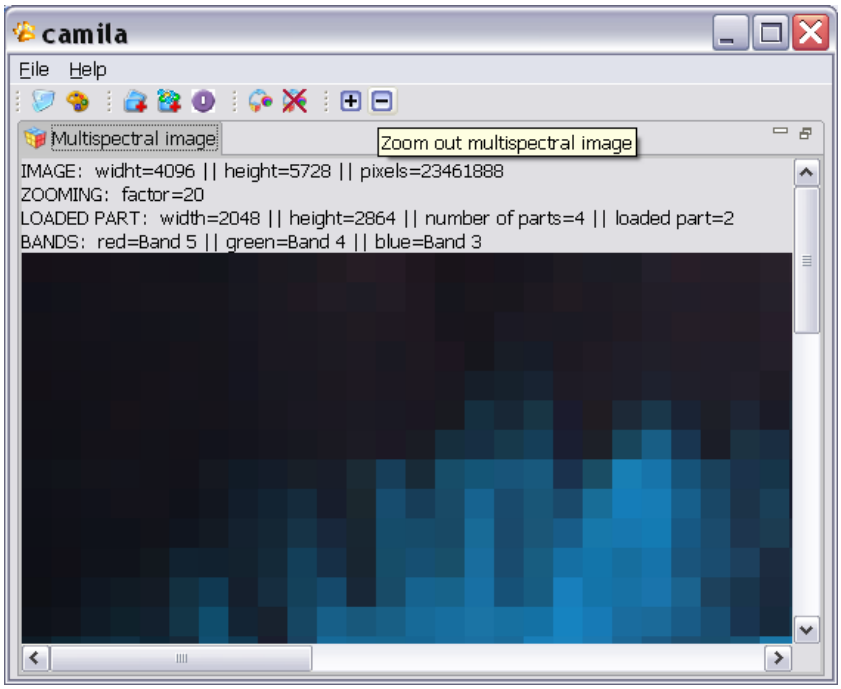

Figura A-20 Escoge “Zoom out multispectral Image”.

Se grafica la imagen original y en su parte superior se presenta información relevante acerca de la imagen y su graficación.

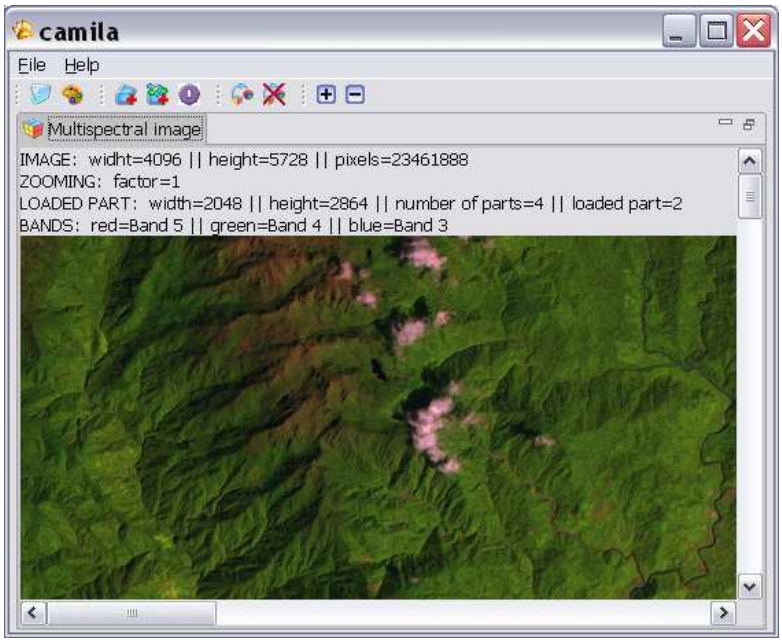

Figura A-21 Imagen original "Zom out multispetral image".

8. CLASIFICAR UNA IMAGEN MULTIESPECTRAL.

Se escoge "Classify multispectral image" en el menú "File" o en el icono de la barra de herramientas.

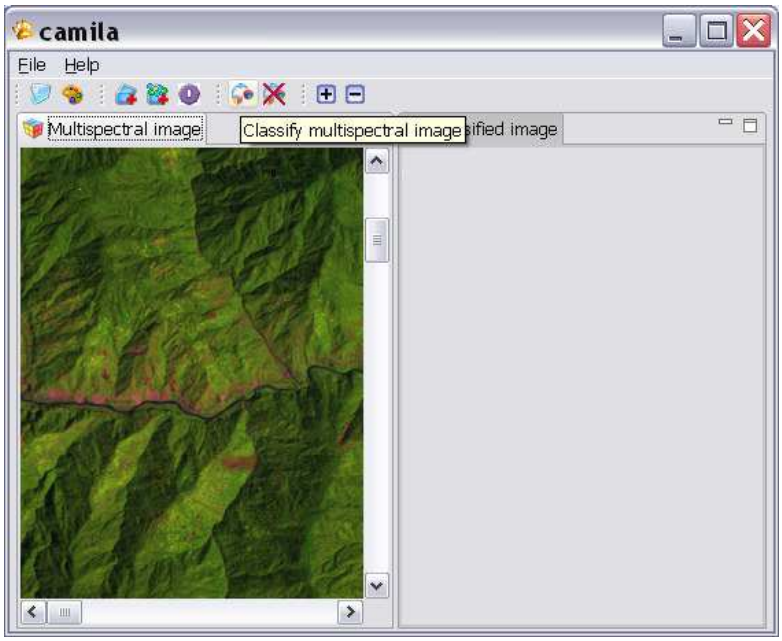

Figura A-22 Escoge "Classify multispectral image".

En el wizard: se escoge el tipo de ejecución a realizar, esta puede ser local o remota (usando el Grid); se escoge el percentil a usar para la pertenencia de cada píxel de la imagen a una clase; y se escribe el número de píxeles estimado a clasificar por job o thread dependiendo si la clasificación es local o usando WS-GRAM.

La clasificación local es usada solo para dar una idea de la diferencia de tiempo que tarda en clasificar toda la imagen local y remotamente. Una clasificación local con muchos píxeles de clases es inconveniente.

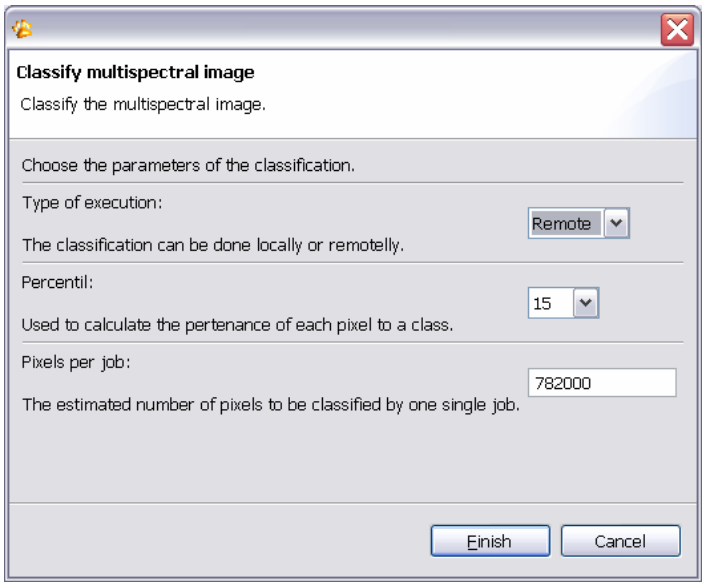

Figura A-23 Wizard "Classify multispectral image".

Se presiona "Finish". CAMILA iniciará el proceso de clasificación en el background, desplegando información acerca del proceso de clasificación en la parte superior de la sección "Classified image".

O

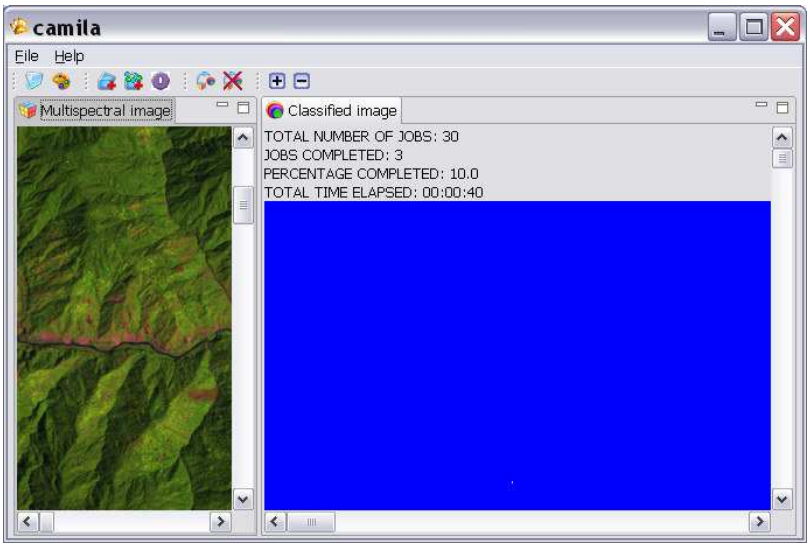

Figura A-24 Clasificando "Cassify multispetral image".

Una vez que la imagen ha sido clasificada se despliega un mensaje informándo del hecho.

Figura A-25 Mensaje clasificando "Cassify multispetral image".

La imagen clasificada presenta en su parte superior información relevante acerca del proceso finalizado de clasificación.

P

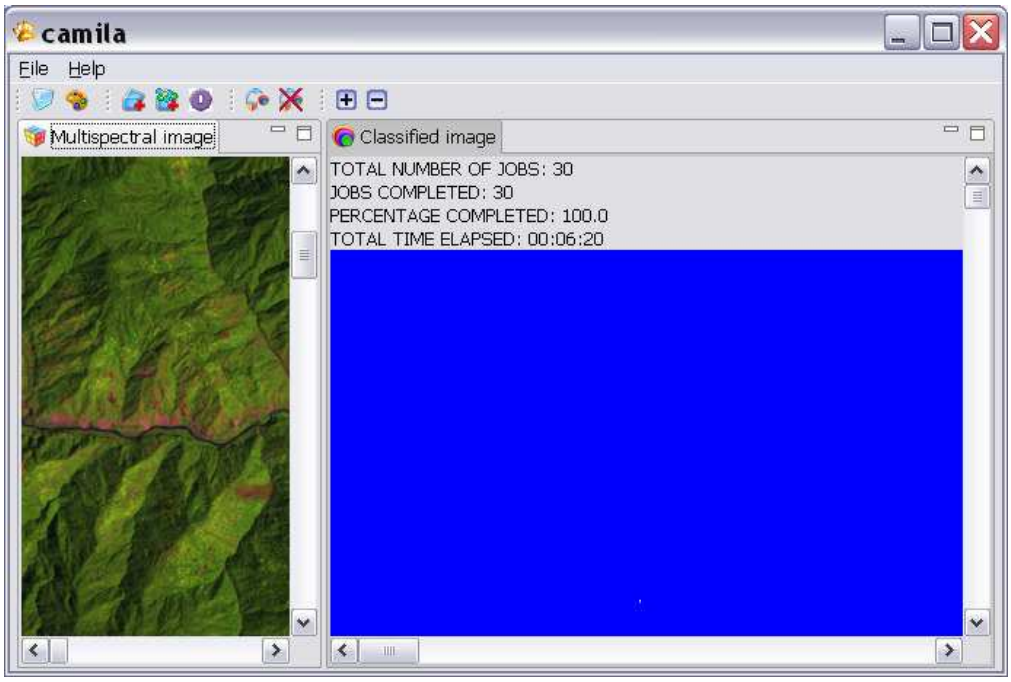

Figura A-26 Imagen clasificada "Cassify multispetral image".

9. CANCELAR LA CLASIFICACIÓN.

Se escoge "Cancel classification" en el menú "File" o en el icono de la barra de herramientas.

Q

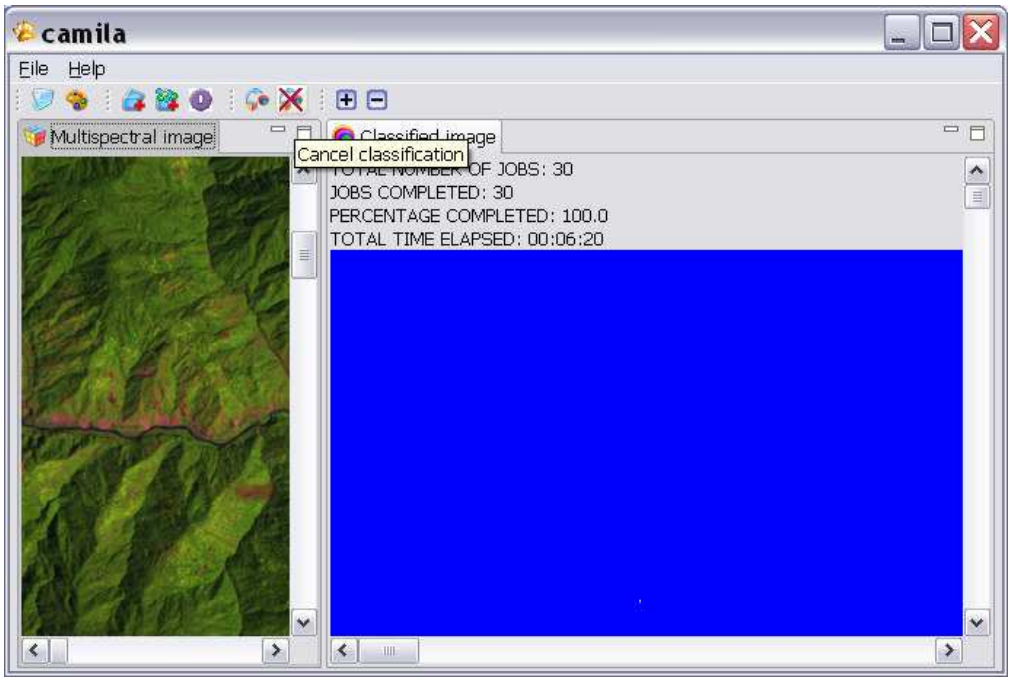

Figura A-27 Escoge "Cancel classification".

Se muestra un mensaje diciendo que los jobs pendientes han sido cancelados. La cancelación en los nodos de ejecución toma tiempo, después de haberse mostrado el mensaje.

Figura A-28 Mensaje cancelar "Cancel classification".

ANEXO No. 2 MANUAL DE INSTALACIÓN Y CONFIGURACIÓN DEL GRID

Este manual muestra la instalación de los servidores que se usaron para la demostración de la aplicación Grid. Muestra la instalación para Globus Toolkit 4.0.3 sobre Debian Sarge 3.1.
La instalación es igual en todos los servidores y adicionalmente en uno de ellos se instaló la autoridad certificadora SimpleCA.

La instalación y configuración consta de las siguientes tareas:

- Instalación y configuración de prerrequisitos.
- Instalación del toolkit.
- Configuración de SimpleCA.
- Creación de certificados.
- Configuración de servicios.

1. INSTALACIÓN Y CONFIGURACIÓN DE PRERREQUISITOS

1.1. Prerrequisitos de software

- jdk-1_5_0_08
- apache-ant-1.6.5
- Compilador de C, gcc. (GNU Compiler Collection)
- Compilador de C++, g++.
- GNU tar
- GNU sed
- zlib 1.1.4
- GNU Make
- Perl
- sudo
- PostgreSQL 7.1.4, base de datos que se puede conectar con JDBC.

1.2. Instalación y configuración

1. Se comprueba la existencia de bibliotecas de desarrollo para GSI-OpenSSH.

```
debian02:~# dpkg --list | grep zlib
ii  zlib-bin       1.2.2-4        compression library - sample
programs
ii  zlib1g         1.2.2-4        compression library - runtime
ii  zlib1g-dev     1.2.2-4        compression library - development
```

2. Se instala el jdk-1_5_0_08, ejecutando las siguientes tareas:
   a. Se crea un directorio.

```
debian02:~# mkdir /usr/java
```

   b. Seguidamente se copia el jdk en el directorio creado.

```
debian02:~# cp /media/cdrom0/jdk-1_5_0_08-linux-i586.bin /usr/java
```

   c. Luego se ejecuta

```
debian02:/usr/java# ./jdk-1_5_0_08-linux-i586.bin
```

   d. Finalmente se acepta la licencia y se visualiza la creación de paquetes.

```
Creating jdk1.5.0_08/lib/tools.jar
Creating jdk1.5.0_08/jre/lib/rt.jar
Creating jdk1.5.0_08/jre/lib/jsse.jar
Creating jdk1.5.0_08/jre/lib/charsets.jar
Creating jdk1.5.0_08/jre/lib/ext/localedata.jar
Creating jdk1.5.0_08/jre/lib/plugin.jar
Creating jdk1.5.0_08/jre/lib/javaws.jar
Creating jdk1.5.0_08/jre/lib/deploy.jar

Done.
```

3. Se instala el apache apache-ant-1.6.5, ejecutando las siguientes tareas:
   a. Se copia el apache-ant en el directorio “local”.\

```
debian02:/usr# cp Desktop/apache-ant-1.6.5 /usr/local/
```

   b. Se visualiza los archivos.

```
debian02:/usr# cd local/
debian02:/usr/local# ls apache-ant-1.6.5
bin   INSTALL  LICENSE      LICENSE.xerces  TODO
docs  KEYS     LICENSE.dom  NOTICE          welcome.html
etc   lib      LICENSE.sax  README          WHATSNEW
```

4. Se revisa las versiones de compiladores C.

```
debian02:/usr/local# gcc --version
gcc (GCC) 3.3.5 (Debian 1:3.3.5-13)
Copyright (C) 2003 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There
is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE.
```

5. Se revisa las versiones de compiladores C++.

```
debian02:/usr/local# g++ --version
g++ (GCC) 3.3.5 (Debian 1:3.3.5-13)
Copyright (C) 2003 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There
is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE.
```

6. Se revisa las versiones de GNU tar.

```
debian02:/usr/local# tar --version
tar (GNU tar) 1.14
Copyright (C) 2004 Free Software Foundation, Inc.
This program comes with NO WARRANTY, to the extent permitted by law.
You may redistribute it under the terms of the GNU General Public
License;
see the file named COPYING for details.
Written by John Gilmore and Jay Fenlason.
```

7. Se revisa las versiones de GNU sed.

```
debian02:/usr/local# sed --version
GNU sed version 4.1.2
Copyright (C) 2003 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There
is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE,
to the extent permitted by law.
```

8. Se revisa las versiones de GNU make.

```
debian02:/usr/local# make --version
GNU Make 3.80
Copyright (C) 2002  Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
```

9. Se comprueba la existencia de perl.

```
debian02:/usr/local# perl --version

This is perl, v5.8.4 built for i386-linux-thread-multi

Copyright 1987-2004, Larry Wall

Perl may be copied only under the terms of either the Artistic
License or the
GNU General Public License, which may be found in the Perl 5 source
kit.

Complete documentation for Perl, including FAQ lists, should be
found on
this system using `man perl' or `perldoc perl'.  If you have access
to the
Internet, point your browser at http://www.perl.com/, the Perl Home
Page.
```

10. Se revisa la versión de sudo.

```
debian02:/# sudo -V
Sudo version 1.6.8p7
```

11. Se comprueba la existencia de PostgreSQL 7.1.4, base de datos que se puede conectar con JDBC

```
debian02:/# dpkg --list | grep postgres
ii  postgresql     7.4.7-6sarge1  object-relational SQL database
management sy
ii  postgresql-cli 7.4.7-6sarge1  front-end programs for PostgreSQL
ii  postgresql-con 7.4.7-6sarge1  additional facilities for
PostgreSQL
ii  postgresql-doc 7.4.7-6sarge1  documentation for the PostgreSQL
database ma
```

12. Se instala el IODBC, prerrequisito opcional para RLS

```
debian02:/# aptitude install libiodbc2 libiodbc2-dev
debian02:/# dpkg --list | grep libiodbc
ii  libiodbc2      3.52.2-3       iODBC Driver Manager
ii  libiodbc2-dev  3.52.2-3       iODBC Driver Manager (development
files)
```

2. INSTALACIÓN DEL TOOLKIT

1. Se crea el usuario “globus”.

```
debian02:/# adduser globus
Adding user `globus'...
Adding new group `globus' (1001).
Adding new user `globus' (1001) with group `globus'.
Creating home directory `/home/globus'.
Copying files from `/etc/skel'
Enter new UNIX password:globus
Retype new UNIX password:globus
passwd: password updated successfully
Changing the user information for globus
Enter the new value, or press ENTER for the default
        Full Name []: globus
        Room Number []:
        Work Phone []: 2524826
        Home Phone []: 2524826
        Other []:
Is the information correct? [y/N] y
```

2. Se crea un directorio, que pertenecerá al usuario globus creado y al grupo globus. Es importante que este usuario pueda realizar tareas administrativas, para lo cual se debe añadir a los siguientes grupos: adm, bind y daemon para que la instalación del toolkit se ejecute sin errores.

```
debian:/etc/init.d# mkdir /usr/local/globus-4.0.3
root@debian:/etc/init.d# chown globus:globus /usr/local/globus-
4.0.3/
```

3. Como usuario globus, para construir el toolkit. Se copia el instalador del gt4.0.3 en el directorio creado.

```
root@debian:/etc/init.d# su globus
globus@debian:/$ cp Desktop/gt4.0.3-all-source-installer
/usr/local/globus-4.0.3/
```

4. Se ejecuta el toolkit.

```
globus@debian:/usr/local/globus-4.0.3/gt4.0.3-all-source-installer$
./configure -prefix=/usr/local/globus-4.0.3/ --with-iodbc=/usr/lib/
checking build system type… i686-pc-linux-gnu
checking for javac… /usr/java/jdk1.5.0_08/bin/javac
checking for ant… /usr/local/apache-ant-1.6.5/bin/ant
configure: creating ./config.status
config.status: creating Makefile
```

5. Se compila el toolkit.

```
globus@debian:/usr/local/globus-4.0.3/gt4.0.3-all-source-installer$
make | tee installer.log
gpt-build ==➔ BUILDING FLAVOR gcc32dbg
gpt-build ==➔ Changing to /usr/local/globus-4.0.3/etc
gpt-build ==➔ REMOVING empty package globus_rendezvous-gcc32dbg-pgm
gpt-build ==➔ REMOVING empty package globus_rendezvous-gcc32dbg-
pgm_static
gpt-build ==➔ REMOVING empty package globus_rendezvous-noflavor-
data
gpt-build ==➔ REMOVING empty package globus_rendezvous-noflavor-doc
echo "Your build completed successfully.  Please run make install."
Your build completed successfully.  Please run make install.
```

6. Se instala el toolkit.

```
root@debian02:/usr/local/globus-4.0.3/gt4.0.3-all-source-installer#
make install
config.status: creating fork.pm
..Done
```

7. Finalmente, se setea las variables de ambiente en los archivos /etc/profile y /etc/bash.bashrc, añadiendo las siguientes líneas:

```
.
ANT_HOME=/usr/local/apache-ant-1.6.5
JAVA_HOME=/usr/java/jdk1.5.0_08
GLOBUS_LOCATION=/usr/local/globus-4.0.3
PATH=$ANT_HOME/bin:$JAVA_HOME/bin:$GLOBUS_LOCATION/bin:$PATH

export PATH ANT_HOME JAVA_HOME GLOBUS_LOCATION
```

3. CONFIGURACIÓN DE SIMPLECA

1. Como usuario "globus", se ejecuta el script de instalación. Este script instala la autoridad certificadora y genera un paquete que puede ser

distribuido a los usuarios de la autoridad certificadora. Se deben ejecutar las siguientes tareas:

a. Se acepta el nombre subject para esta autoridad certificadora.
b. Se configura al email de la autoridad certificadora.
c. Se configura la fecha de expiración. Aceptamos la que viene por defecto.
d. Se ingresa el password de los certificados de la autoridad certificadora, el cual será usado cuando se firmen certificados.

```
globus@debian:~$ source $GLOBUS_LOCATION/etc/globus-user-env.sh
globus@debian:~$ $GLOBUS_LOCATION/setup/globus/setup-simple-ca

WARNING: GPT_LOCATION not set, assuming:
         GPT_LOCATION=/usr/local/globus-4.0.3


    C e r t i f i c a t e   A u t h o r i t y   S e t u p

This script will setup a Certificate Authority for signing Globus
users certificates.  It will also generate a simple CA package
that can be distributed to the users of the CA.

The CA information about the certificates it distributes will
be kept in:

/home/globus/.globus/simpleCA/

The unique subject name for this CA is:

cn=Globus Simple CA, ou=simpleCA-debian.sistemas.epn.edu.ec,
ou=GlobusTest, o=Grid

Do you want to keep this as the CA subject (y/n) [y]:y  a

Enter the email of the CA (this is the email where certificate
requests will be sent to be signed by the CA):viviana@debian  b

The CA certificate has an expiration date. Keep in mind that
once the CA certificate has expired, all the certificates
signed by that CA become invalid.  A CA should regenerate
the CA certificate and start re-issuing ca-setup packages
before the actual CA certificate expires.  This can be done
by re-running this setup script.  Enter the number of DAYS
the CA certificate should last before it expires.
[default: 5 years (1825 days)]:  c

Enter PEM pass phrase: sevenmv              d
Verifying - Enter PEM pass phrase: sevenmv


creating CA config package...done.
```

```
A self-signed certificate has been generated
for the Certificate Authority with the subject:

/O=Grid/OU=GlobusTest/OU=simpleCA-
debian.sistemas.epn.edu.ec/CN=Globus Simple CA

If this is invalid, rerun this script

/usr/local/globus-4.0.3/setup/globus/setup-simple-ca

and enter the appropriate fields.

-------------------------------------------------------------------

The private key of the CA is stored in
/home/globus/.globus/simpleCA//private/cakey.pem
The public CA certificate is stored in
/home/globus/.globus/simpleCA//cacert.pem

The distribution package built for this CA is stored in

/home/globus/.globus/simpleCA//globus_simple_ca_71f5b3e4_setup-
0.19.tar.gz

This file must be distributed to any host wishing to request
certificates from this CA.

CA setup complete.

The following commands will now be run to setup the security
configuration files for this CA:

$GLOBUS_LOCATION/sbin/gpt-build
/home/globus/.globus/simpleCA//globus_simple_ca_71f5b3e4_setup-
0.19.tar.gz

$GLOBUS_LOCATION/sbin/gpt-postinstall
-------------------------------------------------------------------


setup-ssl-utils: Configuring ssl-utils package
Running setup-ssl-utils-sh-scripts...

*******************************************************************
*******

Note: To complete setup of the GSI software you need to run the
following script as root to configure your security configuration
directory:

/usr/local/globus-4.0.3/setup/globus_simple_ca_71f5b3e4_setup/setup-
gsi

For further information on using the setup-gsi script, use the -help
option.  The -default option sets this security configuration to be
the default, and -nonroot can be used on systems where root access
is
not available.

*******************************************************************
```

```
*******

setup-ssl-utils: Complete
```

2. Se revisa el paquete generado

```
globus@debian:/$ ls ~/.globus/
persisted  simpleCA
globus@debian:/$ ls ~/.globus/simpleCA/
cacert.pem  globus_simple_ca_71f5b3e4_setup-0.19.tar.gz  newcerts
certs       grid-ca-ssl.conf                             private
crl         index.txt                                    serial
```

3. Se revisa el hash de la autoridad certificadora. La cadena de dígitos hexadecimal, en este caso el número “71f5b3e4”, es conocido como hash de la autoridad certificadora.

```
root@debian:~/.globus/simpleCA# cd ../..
root@debian:~# ls -l /usr/local/globus-4.0.3/setup/
total 12
drwxrwxrwx  3 root globus 4096 2007-01-04 13:40 globus
drwxr-xr-x  2 root root   4096 2007-01-04 13:40
globus_simple_ca_71f5b3e4_setup
drwxrwxrwx  2 root globus 4096 2006-12-08 09:52 gsi_openssh_setup
```

4. Para que el servidor confíe en la nueva autoridad certificadora. Se logea como usuario root y se ejecuta el siguiente comando:

```
globus@debian:~$ su root
Password:
debian:/home/globus# cd ../..
debian:/# /usr/local/globus-
4.0.3/setup/globus_simple_ca_71f5b3e4_setup/setup-gsi -default
setup-gsi: Configuring GSI security
Installing /etc/grid-security/certificates//grid-
security.conf.71f5b3e4...
Running grid-security-config...
Installing Globus CA certificate into trusted CA certificate
directory...
Installing Globus CA signing policy into trusted CA certificate
directory...
setup-gsi: Complete
debian:/#
```

5. Se revisan los archivos de configuración que establecen la confianza para el SimpleCA. Es importante notar que el valor del hash de estos archivos “71f5b3e4” encaja con el valor del hash del simpleCA.

```
debian:/# ls /etc/grid-security/
certificates  globus-host-ssl.conf  globus-user-ssl.conf  grid-
security.conf
debian:/# ls /etc/grid-security/certificates/
71f5b3e4.0               globus-host-ssl.conf.71f5b3e4  grid-
security.conf.71f5b3e4
71f5b3e4.signing_policy  globus-user-ssl.conf.71f5b3e4
debian:/#
```

4. CREACIÓN DE CERTIFICADOS.

4.1. Certificado de host

1. Como usuario root, se verifica el nombre del servidor

```
debian:/# hostname
debian.sistemas.epn.edu.ec
```

2. Se setea la variable de ambiente y se solicita el certificado de host para el servidor.

```
debian:/# source $GLOBUS_LOCATION/etc/globus-user-env.sh
debian:/# grid-cert-request -host debian.sistemas.epn.edu.ec
Generating a 1024 bit RSA private key
.........................++++++
.++++++
writing new private key to '/etc/grid-security/hostkey.pem'
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Level 0 Organization [Grid]:Level 0 Organizational Unit
[GlobusTest]:Level 1 Organizational Unit [simpleCA-
debian.sistemas.epn.edu.ec]:Name (e.g., John M. Smith) []:

A private host key and a certificate request has been generated
with the subject:

/O=Grid/OU=GlobusTest/OU=simpleCA-
debian.sistemas.epn.edu.ec/CN=host/debian.sistemas.epn.edu.ec

------------------------------------------------------------

The private key is stored in /etc/grid-security/hostkey.pem
The request is stored in /etc/grid-security/hostcert_request.pem

Please e-mail the request to the Globus Simple CA viviana@debian
You may use a command similar to the following:

 cat /etc/grid-security/hostcert_request.pem | mail viviana@debian

Only use the above if this machine can send AND receive e-mail. if
not, please
mail using some other method.

Your certificate will be mailed to you within two working days.
If you receive no response, contact Globus Simple CA at
viviana@debian
debian:/#
```

3. Como usuario globus, se genera el certificado firmado "hostsigned.pem". Si se genera un error, se puede crear manualmente el certificado "hostsigned.pem" copiando y remombrando en nuevo certificado "01.pem".

```
globus@debian:/$ grid-ca-sign -in /etc/grid-
security/hostcert_request.pem -out hostsigned.pem

To sign the request
please enter the password for the CA key:

The new signed certificate is at:
/home/globus/.globus/simpleCA//newcerts/01.pem
```

4. Se verifica la existencia del certificado firmado.

```
globus@debian:~$ ls -l
total 4
-rw-r--r--  1 globus globus 2767 2007-01-15 12:45 hostsigned.pem
```

5. Como usuario root, copiamos el certificado firmado "hostsigned.pem" al directorio generado cuando se solicita un certificado de host (/etc/grid-security).

```
globus@debian:/$ su root
Password:
debian:/# cp ~globus/hostsigned.pem /etc/grid-security/hostcert.pem
```

6. Los certificados hostcert and hostkey son propios del root(se usaran por el servidor GridFTP). El contenedor web necesita estos certificados, para ser ejecutados como usuario no root. Debido a esto creamos los archivos containercert.pem y containerkey.pem con propietario globus en base a los certificados hostcert and hostkey

```
debian:/etc/grid-security# cp hostcert.pem containercert.pem
debian:/etc/grid-security# cp hostkey.pem containerkey.pem
debian:/etc/grid-security# chown globus:globus container*.pem
debian:/etc/grid-security# ls -l *.pem
-rw-r--r--  1 globus globus 2767 2007-01-15 12:49 containercert.pem
-r--------  1 globus globus  887 2007-01-15 12:49 containerkey.pem
-rw-r--r--  1 root   root   2767 2007-01-15 12:48 hostcert.pem
-rw-r--r--  1 root   root   1444 2007-01-15 12:36
hostcert_request.pem
-r--------  1 root   root    887 2007-01-15 12:36 hostkey.pem
debian:/etc/grid-security#
```

7. Como usuario root, se verifica que el certificado de host este creado correctamente.

```
root@debian:/# openssl verify -CApath /etc/grid-
security/certificates/ -purpose sslserver /etc/grid-
security/hostcert.pem
/etc/grid-security/hostcert.pem: OK
```

4.2. Certificado de usuario

1. Con una cuenta local. En nuestro caso con la cuenta de usuario viviana, se setea la variable de ambiente y se solicita el certificado de usuario.

```
debian:~# su viviana
viviana@debian:/root$ cd ..
viviana@debian:/$ source $GLOBUS_LOCATION/etc/globus-user-env.sh
viviana@debian:/$ grid-cert-request
A certificate request and private key is being created.
You will be asked to enter a PEM pass phrase.
This pass phrase is akin to your account password,
and is used to protect your key file.
If you forget your pass phrase, you will need to
obtain a new certificate.

Generating a 1024 bit RSA private key
...................++++++
...............++++++
writing new private key to '/home/viviana/.globus/userkey.pem'
Enter PEM pass phrase: sevenmv
Verifying - Enter PEM pass phrase: sevenmv
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Level 0 Organization [Grid]:Level 0 Organizational Unit
[GlobusTest]:Level 1 Org anizational Unit [simpleCA-
debian.sistemas.epn.edu.ec]:Level 2 Organizational Un it
[sistemas.epn.edu.ec]:Name (e.g., John M. Smith) []:

A private key and a certificate request has been generated with the
subject:

/O=Grid/OU=GlobusTest/OU=simpleCA-
debian.sistemas.epn.edu.ec/OU=sistemas.epn.edu .ec/CN=Viviana Caiza

If the CN=Viviana Caiza is not appropriate, rerun this
script with the -force -cn "Common Name" options.

Your private key is stored in /home/viviana/.globus/userkey.pem
Your request is stored in /home/viviana/.globus/usercert_request.pem

Please e-mail the request to the Globus Simple CA viviana@debian
You may use a command similar to the following:

  cat /home/viviana/.globus/usercert_request.pem | mail
viviana@debian

Only use the above if this machine can send AND receive e-mail. if
not, please
```

```
mail using some other method.

Your certificate will be mailed to you within two working days.
If you receive no response, contact Globus Simple CA at
viviana@debian
viviana@debian:/$
```

2. Como usuario globus, se genera el certificado firmado “signed.pem”. Si se genera un error, se puede crear manualmente el certificado “signed.pem” copiando y remombrando en nuevo certificado “02.pem”.

```
globus@pos02:/$ cd ~
globus@pos02:~$ grid-ca-sign -in
/home/viviana/.globus/usercert_request.pem -out signed.pem

To sign the request
please enter the password for the CA key:

The new signed certificate is at:
/home/globus/.globus/simpleCA//newcerts/02.pem
```

3. Como usuario viviana, se copia el certificado firmado “signed.pem” al directorio generado cuando se solicita un certificado de usuario (~/.globus/).

```
viviana@debian:/$ cp /home/globus/signed.pem ~/.globus/usercert.pem
viviana@debian:/$ ls -l ~/.globus/
total 12
-rw-r--r--  1 viviana viviana 2789 2007-01-16 07:48 usercert.pem
-rw-r--r--  1 viviana viviana 1472 2007-01-16 07:23
usercert_request.pem
-r--------  1 viviana viviana  963 2007-01-16 07:23 userkey.pem
viviana@debian:/$
```

4. Como usuario viviana, se verifica que el certificado de usuario esta creado correctamente.

```
viviana@debian:/$ openssl verify -CApath /etc/grid-
security/certificates/ -purpose sslclient ~/.globus/usercert.pem
/home/viviana/.globus/usercert.pem: OK
```

5. Para obtener autorización, como usuario root, se crea el archivo grid map file.

```
debian:/etc/grid-security# vim /etc/grid-security/grid-mapfile
debian:/etc/grid-security# cat /etc/grid-security/grid-mapfile
"/O=Grid/OU=GlobusTest/OU=simpleCA-
debian.sistemas.epn.edu.ec/OU=sistemas.epn.edu.ec/CN=Viviana Caiza"
viviana
debian:/etc/grid-security#
```

6. se verifica que el archivo grid map file se ha creado correctamente.

```
debian:/etc/grid-security# /usr/local/globus-4.0.3/sbin/grid-
mapfile-check-consistency
Checking /etc/grid-security/grid-mapfile grid mapfile
```

```
Verifying grid mapfile existence...OK
Checking for duplicate entries...OK
Checking for valid user names...OK
```

5. CONFIGURACIÓN DE SERVICIOS.

5.1. GridFTP

1. Como usuario root se configura una entrada en xinetd. Se crea el archivo gridftp con el siguiente contenido: el nombre del servicio usado es gsiftp, usuario root, seteo de variables de ambiente, comando para iniciar el servidor, la opción de linea de comandos –i que configura al servidor GridFTP para que corra bajo xinetd.

```
debian:/etc/grid-security# vim /etc/xinetd.d/gridftp
debian:/etc/grid-security# cat /etc/xinetd.d/gridftp
service gsiftp
{
instances               = 100
socket_type             = stream
wait                    = no
user                    = root
env                     += GLOBUS_LOCATION=/usr/local/globus-4.0.3
env                     += LD_LIBRARY_PATH=/usr/local/globus-
4.0.3/lib
server                  = /usr/local/globus-4.0.3/sbin/globus-
gridftp-server
server_args             = -i
log_on_success          += DURATION
nice                    = 10
disable                 = no
}
```

2. Se crea una entrada en el archivo /etc/services, poniendo el nombre del servicio “gsiftp” con el puerto 2811.

```
root@debian:/etc/grid-security# vim /etc/services

vboxd           20012/udp
binkp           24554/tcp                       # binkp fidonet
protocol
asp             27374/tcp                       # Address Search
Protocol
asp             27374/udp
dircproxy       57000/tcp                       # Detachable IRC
Proxy
tfido           60177/tcp                       # fidonet EMSI over
telnet
fido            60179/tcp                       # fidonet EMSI over
TCP

# Local services
gsiftp          2811/tcp
```

```
root@debian:/etc/grid-security# tail /etc/services
```

3. Se reinicia el daemon xinetd.

```
root@debian02:/etc/grid-security# /etc/init.d/xinetd reload
Reloading internet superserver configuration: xinetd.
```

4. Se verifica que el Puerto 2811 este en escucha.

```
root@debian02:/etc/grid-security# netstat -an | grep 2811
tcp        0      0 0.0.0.0:2811            0.0.0.0:*
LISTEN
```

5. Ahora que el servidor GridFTP está esperando por una solicitud, con una cuenta de usuario (en nuestro caso con el usuario viviana):
    a. Se crea un Proxy.

```
viviana@debian:/$ grid-proxy-init -verify -debug

User Cert File: /home/viviana/.globus/usercert.pem
User Key File: /home/viviana/.globus/userkey.pem

Trusted CA Cert Dir: /etc/grid-security/certificates

Output File: /tmp/x509up_u1002
Your identity: /O=Grid/OU=GlobusTest/OU=simpleCA-
debian.sistemas.epn.edu.ec/OU=sistemas.epn.edu.ec/CN=Viviana Caiza
Enter GRID pass phrase for this identity:
Creating proxy .....................................+++++++++++
................++++++++++++
 Done
Proxy Verify OK
Your proxy is valid until: Tue Jan 16 21:09:30 2007
```

    b. Se verifica la fecha se expiración del Proxy creado.

```
viviana@debian:/$ grid-cert-info
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4 (0x4)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: O=Grid, OU=GlobusTest, OU=simpleCA-
debian.sistemas.epn.edu.ec, CN=Globus Simple CA
        Validity
            Not Before: Jan 16 12:30:05 2007 GMT
            Not After : Jan 16 12:30:05 2008 GMT
        Subject: O=Grid, OU=GlobusTest, OU=simpleCA-
debian.sistemas.epn.edu.ec, OU=sistemas.epn.edu.ec, CN=Viviana Caiza
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:b0:5c:ec:51:75:d1:14:f4:c7:db:6b:ab:cf:ea:
                    e1:db:09:69:92:98:60:5f:83:e9:73:d4:cb:cc:6c:
                    96:d7:8a:41:f4:38:ff:0d:33:9d:07:bd:66:bd:63:
                    e6:cc:23:e4:f2:88:bd:86:08:22:da:8b:4f:65:25:
                    a0:c7:bb:3b:de:97:9b:b1:5e:38:04:d1:b3:c1:10:
                    80:2d:a4:4c:43:c7:28:5c:51:0b:fa:4e:72:e6:ea:
```

```
                    8b:c9:c0:66:14:62:2c:1c:58:5e:00:ae:c6:46:bf:
                    fd:0f:34:38:57:c3:8b:f2:f1:43:3a:39:8e:18:3e:
                    fe:25:fb:80:c0:82:84:35:f3
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            Netscape Cert Type:
                SSL Client, SSL Server, S/MIME, Object Signing
    Signature Algorithm: md5WithRSAEncryption
        ce:d5:f3:e6:1d:ac:2d:1f:39:ca:34:4d:8a:dc:ae:da:42:22:
        97:e7:18:c9:f3:3f:da:9b:03:21:74:06:0c:bf:62:39:8d:3b:
        cc:c4:3a:c0:77:09:5e:ff:21:04:6c:59:7b:09:44:2c:bf:bf:
        27:b9:68:c4:e3:01:ae:20:08:e2:4a:5f:73:6c:43:5b:ee:3d:
        fe:4b:eb:29:1c:2e:6b:34:4e:fe:e6:b5:7d:08:05:04:58:05:
        52:84:24:96:3e:f0:16:e5:b2:11:7d:c8:96:9d:20:6d:6f:c1:
        22:cf:98:9d:fb:51:f5:e0:dc:3a:e8:88:2e:17:94:af:7f:ff:
        04:dd
```

c. Se corre un cliente y se transfiere un archivo.

```
viviana@debian:/$ globus-url-copy
gsiftp://debian.sistemas.epn.edu.ec/etc/group
file:///tmp/viviana.test.copy
```

d. Se comprueba que el archivo se ha transferido.

```
viviana@debian:/$ cd ///tmp
viviana@debian:/tmp$ ls -l
total 24
drwx------  3 root    root    4096 2007-01-16 07:17 gconfd-root
drwx------  2 root    root    4096 2007-01-16 07:17 keyring-0e3O9g
srwxr-xr-x  1 root    root       0 2007-01-16 07:17 mapping-root
drwx------  2 root    root    4096 2007-01-16 09:08 orbit-root
drwx------  2 root    root    4096 2007-01-16 07:17 ssh-yyukKi1668
-rw-r--r--  1 viviana viviana  795 2007-01-16 09:14
viviana.test.copy
-rw-------  1 viviana viviana 2349 2007-01-16 09:09 x509up_u1002
```

e. Se compara el archivo “viviana.test.copy” con el archivo “group”, linea por linea.

```
viviana@debian:/tmp$ diff viviana.test.copy /etc/group
```

5.2. Contenedor de Web Services

1. Como usuario globus, se crea el scipt start-stop con el seteo las variables y programas para iniciar o detener el contenedor.

```
globus@debian:/$ cat $GLOBUS_LOCATION/start-stop
#! /bin/sh
set -e
export GLOBUS_LOCATION=/usr/local/globus-4.0.3
export JAVA_HOME=/usr/java/jdk1.5.0_08/
export ANT_HOME=/usr/local/apache-ant-1.6.5
export GLOBUS_OPTIONS="-Xms256M -Xmx512M"
. $GLOBUS_LOCATION/etc/globus-user-env.sh
cd $GLOBUS_LOCATION
case "$1" in
    start)
```

```
        $GLOBUS_LOCATION/sbin/globus-start-container-detached -p
8443
        ;;
    stop)
        $GLOBUS_LOCATION/sbin/globus-stop-container-detached
        ;;
    *)
        echo "Usage: globus {start|stop}" >&2
        exit 1
        ;;
esac
exit 0
```

2. Al usuario globus se le asigna permiso de ejecución al script “start-stop” creado.

```
globus@debian:/$ chmod +x $GLOBUS_LOCATION/start-stop
```

3. Como usuario root, Se crea el script “globus-4.0.3” en el /etc/init.d/, para llamar al script “start-stop” como usuario globus.

```
debian:~# vim /etc/init.d/globus-4.0.3
debian:~# cat /etc/init.d/globus-4.0.3
#!/bin/sh -e
case "$1" in
  start)
    su - globus /usr/local/globus-4.0.3/start-stop start
    ;;
  stop)
    su - globus /usr/local/globus-4.0.3/start-stop stop
    ;;
  restart)
    $0 stop
    sleep 1
    $0 start
    ;;
  *)
    printf "Usage: $0 {start|stop|restart}\n" >&2
    exit 1
    ;;
esac
exit 0
```

4. Al usuario root se le asigna permiso de ejecución al script “globus-4.0.3” creado.

```
debian:~# chmod +x /etc/init.d/globus-4.0.3
```

5. Como usuario root se ejecuta el script “globus-4.0.3”, el mismo que inicia el script “start-stop” como usuario globus, el cual a su vez iniciara el contenedor de web services por el puerto 8443. Si ocurriera algún error se debe revisar que existan permisos de lectura para el usuario globus en los archivos server-config.wsdd y jndi-config.xml.

```
debian:/# /etc/init.d/globus-4.0.3 start
Starting Globus container. PID: 2612
debian:/#
```

6. Se reviza el log. Si existen errores en el log se debe a que aun no se tiene configurado el RFT.

```
debian:/# cat $GLOBUS_LOCATION/var/container.log
```

7. Para interactuar con el contenedor. Con una cuenta de usuario (en nuestro caso la cuenta viviana)se debe realizar lo siguiente:
    a. Se crea un Proxy o se verifica que exista uno.

```
viviana@debian:/$ grid-proxy-init -verify -debug

User Cert File: /home/viviana/.globus/usercert.pem
User Key File: /home/viviana/.globus/userkey.pem

Trusted CA Cert Dir: /etc/grid-security/certificates

Output File: /tmp/x509up_u1002
Your identity: /O=Grid/OU=GlobusTest/OU=simpleCA-
debian.sistemas.epn.edu.ec/OU=sistemas.epn.edu.ec/CN=Viviana Caiza
Enter GRID pass phrase for this identity:
Creating proxy ..+++++++++++
.......++++++++++++
 Done
Proxy Verify OK
Your proxy is valid until: Wed Jan 17 00:30:31 2007
```

    b. Se ejecuta el comando.

```
viviana@debian:/$ counter-client -s
https://debian.sistemas.epn.edu.ec:8443/wsrf/services/CounterService
Got notification with value: 3
Counter has value: 3
Got notification with value: 13
```

5.3. RFT

Una vez instalado y configurado un contenedor de web services, un certificado de host, un servidor GridFTP y la base de datos postgreSQL con interfaz JDBC, para almacenar el estado de la transferencia.

1. Como usuario root, se configura la base de datos postgreSQL.
    a. Se configura el daemon postmaster para que acepte conecciones TCP, añadiendo la opción POSTMASTER_OPTIONS="-i".

```
root@debian:/# vim /etc/postgresql/postmaster.conf
root@debian:/# grep POSTMASTER /etc/postgresql/postmaster.conf
POSTMASTER_OPTIONS="-i"
```

    b. Se setea la seguridad para la base de datos a crear “rftDatabase” usando la encriptación md5.

```
root@debian:/# vim /etc/postgresql/pg_hba.conf
debian:/# grep rftDatabase /etc/postgresql/pg_hba.conf
host rftDatabase "globus" "192.168.57.32" 255.255.255.0 md5
```

c. Se reinicia el daemon postgreSQL

```
root@debian:/# /etc/init.d/postgresql restart
Stopping PostgreSQL database server: autovacuum postmaster.
Starting PostgreSQL database server: postmaster autovacuum.
```

d. Se necesita crear una cuenta de usuario PostgreSQL (en nuestro caso se crea al usuario globus) para conectarse a la base de datos. Esta cuenta es usualmente bajo la cual corre el contenedor.

```
root@debian:/# su postgres -c "createuser -P globus"
Enter password for new user:globus
Enter it again:globus
Shall the new user be allowed to create databases? (y/n) y
Shall the new user be allowed to create more new users? (y/n) n
CREATE USER
```

e. Con la cuenta de usuario postgreSQL “globus”, se crea la base de datos que se usará para almacenar los estados de transferencias de RFT.

```
root@debian:/# su globus
globus@debian:/$ createdb rftDatabase
CREATE DATABASE
```

f. Para poblar la base de datos RFT con esquemas apropiados, se ejecuta:

```
globus@debian:/$ psql -d rftDatabase -f
$GLOBUS_LOCATION/share/globus_wsrf_rft/rft_schema.sql
psql:/usr/local/globus-4.0.3/share/globus_wsrf_rft/rft_schema.sql:6:
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index
"requestid_pkey" for table "requestid"
CREATE TABLE
psql:/usr/local/globus-
4.0.3/share/globus_wsrf_rft/rft_schema.sql:11: NOTICE:  CREATE TABLE
/ PRIMARY KEY will create implicit index "transferid_pkey" for table
"transferid"
CREATE TABLE
psql:/usr/local/globus-
4.0.3/share/globus_wsrf_rft/rft_schema.sql:30: NOTICE:  CREATE TABLE
/ PRIMARY KEY will create implicit index "request_pkey" for table
"request"
CREATE TABLE
psql:/usr/local/globus-
4.0.3/share/globus_wsrf_rft/rft_schema.sql:65: NOTICE:  CREATE TABLE
/ PRIMARY KEY will create implicit index "transfer_pkey" for table
"transfer"
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE INDEX
globus@debian:/$
```

g. Se configura RFT para encontrar la base de datos. El archivo jndi-config.xml contiene la sección dbConfiguration donde se puede cambiar el parámetro connectionString,apuntando a la máquina en la cual está instalado el postgreSQL y la sección username, con el nombre del usuario propietario de la base de datos y su respectivo password.

```
globus@debian:/$ vim $GLOBUS_LOCATION/etc/globus_wsrf_rft/jndi-
config.xml
globus@debian:/$ grep -C 3 password
$GLOBUS_LOCATION/etc/globus_wsrf_rft/jndi-config.xml
            </parameter>
            <parameter>
                <name>
                password
                </name>
                <value>
                globus
```

2. Como usuario root se reinicia el contenedor de servicios web y se verifica que en el log del contenedor ya no existan errores.

```
globus@debian:/$ su root
Password:
debian:/# /etc/init.d/globus-4.0.3 restart
Stopping Globus container. PID: 2612
Container stopped
Starting Globus container. PID: 3532

debian:/# cat /usr/local/globus-4.0.3/var/container.log
2007-01-17 13:02:51,014 INFO  exec.RunQueue [main,initialize:68]
Starting state machine with 18 run queues.
10.
```

3. Con una cuenta de usuario local (en nuestro caso con el usuario viviana) se prueba transferencias rft.
   a. Se copia el archivo “transfer.xfr” al tmp.

```
debian:/# su viviana
viviana@debian:/$ cp /usr/local/globus-
4.0.3/share/globus_wsrf_rft_test/transfer.xfr /tmp/rft.xfr
```

   b. Se edita una transferencia en el archivo rft.xfr. El origen será “gsiftp://debian.sistemas.epn.edu.ec:2811/etc/group” y el destino sera “gsiftp://debian.sistemas.epn.edu.ec:2811/ tmp/rftTest_Done.tmp”.

```
viviana@debian:/$vim /tmp/rft.xfr
true
16000
16000
false
```

```
1
true
1
null
null
false
10
gsiftp://debian.sistemas.epn.edu.ec:2811/etc/group
gsiftp://debian.sistemas.epn.edu.ec:2811/tmp/rftTest_Done.tmp
```

c. Se prueba la transferencia del archivo rft.xfr en el host "debian.sistemas.epn.edu.ec".

```
viviana@debian:/$ rft -h debian.sistemas.epn.edu.ec -f /tmp/rft.xfr
Number of transfers in this request: 1
Subscribed for overall status
Termination time to set: 60 minutes

 Overall status of transfer:
Finished/Active/Failed/Retrying/Pending
0/1/0/0/0

 Overall status of transfer:
Finished/Active/Failed/Retrying/Pending
1/0/0/0/0
All Transfers are completed
```

d. Se verifica la transferencia de archivos.

```
viviana@debian:/$ cd /tmp/
viviana@debian:/tmp$ ls
gconfd-root        hsperfdata_viviana  orbit-root
rftTest_Done.tmp  x509up_u1002
hsperfdata_globus  keyring-RSTqNc      pruebaHecha.tmp  rft.xfr
hsperfdata_root    mapping-root        prueba.txt       ssh-
eAezPN1666
```

5.4. WS-GRAM

Una vez que se ha configurado GridFTP y RFT, se puede configurar el GRAM para administrar recursos.

1. Se configura el archivo sudores, para que el usuario globus pueda iniciar jobs como un usuario diferente.

```
root@choate:~# visudo
root@choate:~# cat /etc/sudoers
globus ALL=(viviana) NOPASSWD: /usr/local/globus-
4.0.1/libexec/globus-gridmap-and-execute
-g /etc/grid-security/grid-mapfile /usr/local/globus-
4.0.1/libexec/globus-job-manager-script.pl *
globus  ALL=(viviana) NOPASSWD: /usr/local/globus-
4.0.1/libexec/globus-gridmap-and-execute
-g /etc/grid-security/grid-mapfile /usr/local/globus-
4.0.1/libexec/globus-gram-local-proxy-tool *
```

2. Con una cuenta de usuario local (en nuestro caso usuario viviana), con un Proxy valido y con el contenedor iniciado, se prueba el envío de jobs. Para el envio de job con staging. Se copia el comando /bin/echo desde debian a mi directorio home y se lo nombra como my_echo. Seguidamente se ejecuta el my_echo con algunos arcgumentos, y se captura el stderr/stdout. Este envio de jobs es uno de los más caracteristicos que es usado por el servicio RFT en debian para transferir un archivo via el servidor GridFTP.

   a. Se crea un archivo para envío de job staging.

```
viviana@debian:/$ vim a.rsl
viviana@debian:/$ cat a.rsl
<job>
    <executable>my_echo</executable>
    <directory>${GLOBUS_USER_HOME}</directory>
    <argument>Hello</argument>
    <argument>World!</argument>
    <stdout>${GLOBUS_USER_HOME}/stdout</stdout>
    <stderr>${GLOBUS_USER_HOME}/stderr</stderr>
    <fileStageIn>
        <transfer>
<sourceUrl>gsiftp://debian.sistemas.epn.edu.ec:2811/bin/echo</source
Url>
<destinationUrl>file:///${GLOBUS_USER_HOME}/my_echo</destinationUrl>
         </transfer>
    </fileStageIn>
    <fileCleanUp>
         <deletion>
            <file>file:///${GLOBUS_USER_HOME}/my_echo</file>
         </deletion>
    </fileCleanUp>
</job>
```

   b. Se ejecuta el job.

```
viviana@debian:/$ globusrun-ws -submit -S -f a.rsl
Delegating user credentials...Done.
Submitting job...Done.
Job ID: uuid:90045cd8-a66c-11db-a841-001676094f70
Termination time: 01/18/2007 20:51 GMT
Current job state: StageIn
Current job state: Active
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
Cleaning up any delegated credentials...Done.
viviana@debian:/$
```

   c. Se verifica la salida.

```
viviana@debian:/$ cat ~/stdout
Hello World!
viviana@debian:/$ ls ~/my_echo
ls: /home/viviana/my_echo: No such file or directory
viviana@debian:/$
```