Universidad de Castilla-la Mancha

Escuela Superior de Informática

Ingeniería en Informática

Anteproyecto

Geo-Cloud

Modelling and implementation of the Geo-Cloud experiment for the Fed4FIRE European Project

Autor: Rubén Pérez Pascual

Director: Jonathan Becedas

Tutor: Carlos Gonzáles Morcillo

18 de febrero de 2014

Índice

1. Introducción **1**

1.1. Rendering 2

1.2. Problemática 3

2. Objetivos **4**

3. Métodos y fases de trabajo **5**

4. Medios que se pretenden utilizar **6**

4.1. Recursos Hardware 7

4.2. Recursos Software 7

4.3. Documentación 8

1. Introducción

Las compañías dedicadas a la observación de la Tierra (EO industries) implementan la adquisición, almacenamiento, procesamiento y distribución de las imágenes captadas, en su propias granjas de servidores. Sin embargo estas soluciones tienen riesgos ligados a esa infraestructura. Por ejemplo, la escalabilidad. Puede ser que en un momento crítico, se demanden más recursos y esa infraestructura no puede procesar todas las peticiones.

En los últimos años, se ha puesto de moda hablar de *Cloud Computing*. Este concepto se basa en el concepto de procesamiento distribuido. Esto no es ni más ni menos que unificar granjas de servidores para unificar recursos tanto de proceso como de almacenamiento, de forma transparente al usario.

El consorcio europeo Fed4FIRE [**?**] proporciona una serie de *Testbeds* para el desarrollo de distintas investigaciones. Estas herramientas han sido el resultado de varios proyectos europeos y los que cabe destacar bajo el punto de vista de este proyecto son los siguientes:

- *PlanetLab Europe:*

La observación de la Tierra ha sido siempre y será uno de los sectores más Actualmente, uno de los sectores de la informática que más se ha desarrollado ha sido la computación gráfica. Esto es debido a que no solamente ha formado parte de la comunidad científica, si no que ha invadido diversos mercados. Entre estos, se encuentran la visualización médica, el avance de los videojuegos, las cinematográficas y al potente hardware donde desarrollarse. Sobre todo el negocio cinematográfico y de videojuegos están consiguiendo que las técnicas de computación gráfica avancen más rápidamente.

El proceso de construcción de una escena 3D cuenta con una serie de etapas principales que pueden resumirse como:

- *Modelado:* Construcción de la escena representando los objetos con diversas técnicas, ya sean mallas poligonales, superficies NURBS, superficies implícitas, etc...
- *Asignación de materiales:* Cada objeto tiene unas propiedades y comportamientos respecto a como refleja la luz incidente, como el modelo de sombreado empleado, la textura,

color o índice de refracción.

- *Iluminación:*En esta fase se definen las luces de la escena.
- *Fase de rendering:* Dependiendo del nivel de realismo con el que se simule el comportamiento físico de la luz, tenemos diferentes familias de métodos de rendering como los basados en RayCasting, RayTracing, PathTracing, etc...

1.1. Rendering

Antes se mencionó que el último paso para representar una escena 3D como una imagen 2D se denomina rendering. A continuación se explicarán algunos de los algoritmos más usados en este proceso, pero antes se definirán varios conceptos básicos[**?**] para facilitar su comprensión.

- *Plano de vista:* Matriz de píxeles, la cual coincide con el límite de la escena que el observador es capaz de ver desde su posición.
- *Punto de vista:* Situación del observador.

Existen diversos algoritmos con esta finalidad, cada uno con sus ventajas e inconvenientes[**?**]. Estos son *RayCasting*, *RayTracing* y *PathTracing* y básicamente realizan el proceso de render de esta foma:

- *RayCasting:* Consiste en lanzar rayos desde el plano de vista, uno por cada píxel, y encontrar el punto de intersección más cercano con los objetos de la escena. Su eficiencia es bastante aceptable pero genera resultados con poca calidad debido a que no simula efectos de rebote de la luz entre superficies.
- *RayTracing:* Es un método elegante y sencillo que permite calcular de una forma unificada la reflexión y refracción de la luz, sombras, eliminación de objetos ocultos entre otras características. Existen dos aproximaciones para la implementación de este algoritmo. La primera alternativa se basa en lanzar rayos desde las fuentes emisoras de luz y seguir el camino desde estas hasta el punto de vista (*forward RayTracing*). La otra versión del algoritmo (*backward RayTracing*) donde los rayos parten del plano de

imagen y alcanzan a los objetos de la escena. Este método evita trazar inútilmente rayos que en el caso de que se lanzaran desde las fuentes de luz, muchos no llegarían al plano de vista y por lo tanto no representarían nada, únicamente consumiendo recursos computacionales.

- *PathTracing:* Lanza rayos aleatorios para simular sombras suaves, motion blur y profundidad de campo. Este algoritmo calcula todos los posibles caminos de luz, por lo que consigue un resultado muy realista, pero es muy costoso computacionalmente.

1.2. Problemática

Como se ha podido observar, existen limitaciones importantes a la hora de renderizar. El primero es debido al gran coste computacional. Como se ha mencionado anteriormente, los rayos se refractan, reflejan en los distintos objetos, cada uno de ellos con distintas propiedades. Con lo cual, el comportamiento del rayo no está para nada determinado y puede ser complejo estudiar su comportamiento. Las escenas que se quieren renderizar a un nivel de detalle realista y son muy complejas, requieren varias horas de render. Incluso si este proceso se realiza en un *cluster* muy potente y con la última tecnología en redes de altas prestaciones, se requiere un tiempo de cómputo muy alto. Estas condiciones no son permisibles, por ejemplo en algunos casos de diagnóstico por imagen, cuando la imagen se tiene que ir procesando a tiempo real.

En torno a eso, se tiene el problema de que en una imagen existen diversas zonas de distinta complejidad. Estas partes se podrían renderizar usando diferentes algoritmos sin perder demasiada información visual[**?**]. Con este planteamiento, surge el problema de la división de tareas y asignación balanceada a los distintos nodos procesadores. Con lo cual, se podría dividir esa escena respecto a la complejidad de cada partición y a cada una se le asignaría un algoritmo de render para conseguir ganar en eficiencia sin perder calidad en el resultado.

Relacionado con el cálculo de la complejidad de las distintas zonas de la imagen, se plantean diferentes enfoques. Estos pueden ser por ejemplo las distintas fuentes de luz de la escena, la cantidad de caras de los objetos, la distancia a la que están del plano de vista y

si será conveniente o no aplicarle un tipo de renderizado más óptimo si apenas se percibirá, como también el renderizar algún objeto que desde el punto de vista del observador no será visible.

2. Objetivos

Según lo expuesto en la sección 1, sería deseable disponer de un sistema que simule una constelación de satélites y varias estaciones de tierra con sus respectivas métricas en las comunicaciones. Además, que el procesado de los datos en bruto obtenidos en las estaciones de tierra se realice en la nube y la adquisición de esas imágenes procesadas por parte de los clientes sea real.

Por ello en este proyecto se plantean los siguientes objetivos:

- **Implementación de los servicios basados en Cloud en la plataforma BonFIRE:** Se pretende desarrollar un servicio de procesamiento de datos Raw y creación de orto-imágenes en Cloud. Para esta implementación será necesario conocer la arquitectura de la cadena de procesado que se implementa en Elecnor Deimos. Además se debe de dar soporte a las peticiones de los clientes con lo cuál se deben de implementar servicios de distribución de estos resultados entre los usuarios finales.
- **Implementación de la constelación de satélites, estaciones de tierra y clientes sobre PlanetLab Europe:**Otro aspecto a tener en cuenta serían las métricas. Para esto se usará PlanetLab Europe, con la que conseguiremos simular conexiones de red entre distintos nodos distribuidos por el mundo. A tavés de la implementación del modelo de red en PlanetLab Europe se podrán medir las características de una red real. Se realizarán dos capas en esta implementación. La primera modelará las estaciones de tierra y la constelación; la segunda, los clientes finales accediendo a los servicios Cloud.
- **Implementación de la constelación de satélites, estaciones de tierra y clientes sobre VirtualWall:** Tras realizar la implementación sobre PlanetLab Europe, se conseguirá toda la información de una red real, con lo que se realizará un modelo sobre VirtualWall de acuerdo a esos parámetros. Además esta plataforma permitirá tener

completo control sobre los recursos y todos los factores envueltos en los distintos casos de prueba, con lo que se podrá evaluar su comportamiento.

- **Simulación de distintos escenarios y obtención de resultados:** Tras realizar la implementación de lo anteriormente expuesto, se procederá a realizar uno o varios casos de prueba con los que obtengamos datos para contrastar como actúa este servicio de procesado de obtención de geo-imágenes en la nube frente a la misma implementación sobre un centro de procesamiento de datos.

3. Métodos y fases de trabajo

Para llevar a cabo los objetivos propuestos en este documento se dividirá el proyecto en las siguientes fases:

1. **Estudio del estado del arte:** En esta fase será necesario conocer los distintos testbeds proporcionados por el consorcio *Fed4FIRE* y comprender bien los aspectos relativos a su funcionamiento. Se realizará un estudio para obtener las métricas a tener en cuenta a la hora de la implementación de la simulación. Se realizará un estudio en profundidad sobre las posibilidades que ofrece BonFIRE para la creación y control de los experimentos. Se estudiará la plataforma Virtual Wall para la simulación de nodos. Se estudiará la plataforma PlanetLab Europe para la obtención de las métricas reales y poder ser cercano a la realidad. Se estudiará el centro de proceso de datos de la empresa Elecnor Deimos y más concretamente el núcleo de procesamiento de los datos raw para poder implementarlo en la nube que nos proporciona BonFIRE.

2. **Estudio de la viabilidad:** El proyecto se adaptará realizando pruebas para verificar el correcto funcionamiento del mismo y su puesta en marcha. Se dispondrán de los distintos testbeds y el computador de desarrollo donde poder probar la plataforma. Para el uso óptimo de la aplicación es muy recomendable determinar qué requisitos, tanto hardware como software, serán necesarios.

3. **Análisis de la arquitectura del sistema:** Partiendo de los requisitos más importantes que debe cumplir este proyecto, se realizará un análisis más concreto de los

componentes que formarán la arquitectura teniendo en cuenta los principios de modularización y extensibilidad.

4. **Diseño de la arquitectura del sistema:** En esta fase se intentará concretar, a un nivel más cercano a la implementación, cada una de las clases que conformarán la arquitectura del sistema. Se podrá utilizar un lenguaje específico para este propósito como *UML*. Esta fase se realizará de forma concurrente junto a las fases de desarrollo y pruebas, realimentándose para mejorar el diseño de la arquitectura.

5. **Desarrollo del sistema:** Se dará comienzo a la implementación de todos los módulos que formarán el sistema, en el lenguaje y formato escogido, siguiendo la arquitectura del sistema. Se usará una metodología iterativa incremental para el desarrollo del mismo.

6. **Pruebas del sistema:** Se tratará de ir realizando pruebas tanto de caja negra como de caja blanca de cada una de las clases implementadas. Cuando se termine el sistema, se harán pruebas de integración de todos los componentes y su funcionamiento.

7. **Evaluación de los resultados:**

   Se evaluará la ganancia en eficiencia respecto a un sistema de procesamiento de imágenes raw en un CPD el cual no se beneficie de los beneficios de la ejecución y distribución en la nube.

8. **Documentación del proyecto:** De forma paralela al desarrollo del proyecto se irá generando la documentación del mismo (diagramas de clases, de análisis, requisitos, manual de usuario, étc.) para completarla al final con las conclusiones y las pruebas finales que se realicen.

4. Medios que se pretenden utilizar

El propósito del proyecto es realizar una simulación de un sistema distribuido robusto, modular y altamente escalable. La distribución del proyecto se realizará mediante alguna licencia libre como GPLv3[4].

Se seguirá el *proceso unificado de desarrollo de software* durante la realización del proyecto. Para la notación se usará *UML*[8].

4.1. Recursos Hardware

Se dispondrá de varios *Testbed* proporcionados por el proyecto europeo *Fed4FIRE* los cuales son:

1. *PlanetLab Europe[?]:* es la rama europea del sistema PlanetLab, la red más grande del mundo para realizar experimentos. Tiene servidores distribuidos a lo largo de todo el mundo y se pueden evaluar distintas métricas de red como latencia, ruido, pérdida de paquetes, étc. Esta facility se usará para obtener las medidas reales de los parámetros que se utilizarán en los nodos desarrollados en Virtual Wall y BonFIRE.

2. *Virtual Wall[?]:*Consiste en numerosos nodos físicos (concretamente 100) mediante los cuales se pueden realizar simulaciones. Es totalmente configurable, permitiendo crear la red deseada para el experimento y definir los parámetros de la red.

3. *BonFIRE[?]:* Es una plataforma Cloud orientada a la experimentación. Contiene 7 testbeds en Europa geográficamente distribuidos que ofrecen servicios de almacenamiento, procesamiento y networking de manera heterogénea.

   Además se dispondrá de un computador localizado en *Elecnor Deimos* en Puertollano para el desarrollo y la ejecución de los casos de prueba. El hardware de este constará de un procesador Intel Core i5 a 3,2 MHz, 8 Gb de memoria RAM y 1 Tb de disco duro.

4.2. Recursos Software

Los sistemas operativos, entornos de programación y herramientas que se utilizarán en el desarrollo del proyecto serán Software Libre.

- *Sistema Operativo CentOS*[1].

[1] http://www.centos.org/

- *GNU Make [11]:* herramienta para la generación automática de ejecutables, que se empleará para automatizar cualquier proceso susceptible de ello.
- *GNU Emacs [10]:* editor de textos extensible y configurable, para el desarrollo e implementación del proyecto.
- *Git [2]:* ofrece un sistema de control de versiones distribuido y la posibilidad de llevar ese control fuera de línea, frente a los sistemas de control de versiones centralizados como CVS o Subversion.
- *Python [5]:* lenguaje interpretado multiparadigma y multiplataforma que se usará para programar gran parte de este proyecto. Se usará la versión 2.7.4 para el desarrollo de este proyecto. [2]
- *Ruby [3]:* lenguaje interpretado, de propósito general y orientado a objetos que se usará para la construcción de la plataforma de procesamiento de datos raw en la nube. Se usará la versión 1.9 para el desarrollo de este proyecto.
- *Nepi[1]:* Es una plataforma basada en Python para diseñar y realizar experimentos en plataformas de evaluación de redes como por ejemplo PlanetLab. Se usará para realizar la simulación sobre PlanetLab y obtener los parámetros reales para realizar la simulación.
- *GeoServer [6]:*
- *GeoNetwork[7]:*

4.3. Documentación

- LaTeX [9]: lenguaje de marcado de documentos, utilizado para realizar este documento y, en un futuro, la documentación del proyecto.
- BibTeX: herramienta distribuida con LaTeXpara generar las listas de referencias tanto de este documento como de la memoria del proyecto.

[2] Página web oficial de Python

Referencias

[1] Quereilhac A., Tribino J., Guevgeozian L., and Kouvakas A. Nepi v3.0. 2013.

[2] S. Chacon. *Pro Git*. Apress, 2009.

[3] Ruby Software Community. Ruby Documentation. `https://www.ruby-lang.org/en/documentation/`, última consulta el 16 de Febrero de 2014.

[4] Free Software Foundation. Licencia GPL versión 3 de la Free Software Foundation. `http://www.fsf.org/licensing/licenses/gpl.html`, última consulta el 8 de Noviembre de 2013.

[5] Python Software Foundation. Python v3.4 documentation. última consulta el 8 de Noviembre de 2013.

[6] GeoNetwork. Geonetwork user manual. `http://docs.geoserver.org/stable/en/user/`, última consulta el 16 de Febrero de 2014.

[7] GeoNetwork. Geonetwork user manual. Enero, 30 , 2014.

[8] Object Management Group. Omg unified modeling language, infraestructure. Abril, 5 , 2010.

[9] L. Lamport. *LaTEX: User's Guide & Reference Manual*. Addison-Wesley, 1986.

[10] R. Stallman, A. Universitet, and D. Afdeling. *GNU Emacs manual*. Free Software Foundation, 1993.

[11] R.M. Stallman and R. McGrath. *GNU make*, volume 2002. Free Software Foundation, 1988.