Visualización de cascadas de rayos cósmicos sobre GPUs

Rafael Antonio Laverde Laverde

Universidad Industrial de Santander
Facultad de Físico Mecánicas
Escuela de Ingeniería de Sistemas e Informática
Bucaramanga

2015

Visualización de cascadas de rayos cósmicos sobre GPUs

Autor:
Rafael Antonio Laverde Laverde

Proyecto de Grado para optar por el título de Ingeniero de Sistemas

Director:
Ph.D. Carlos Jaime Barrios Hernandez
Co-director:
Ph.D. Hernán Asorey

Universidad Industrial de Santander
Facultad de Físico Mecánicas
Escuela de Ingeniería de Sistemas e Informática
Bucaramanga
2015

Quisiera dedicar este logro a mis padres, mi familia, mis amigos, profesores, y todos aquellos que me acompañaron durante esta importante etapa de mi vida.

AGRADECIMIENTOS

Deseo expresar un especial agradecimiento a:

Al Centro de Supercomputación y Cálculo Científico de la Universidad Industrial de Santander (SC3UIS), por el soporte en el uso de la plataforma *GRIDUIS2.* [1] La cual se utilizo para las pruebas realizadas, y se utilizara para el uso de la aplicación.

Al grupo Halley de Astronomía y Ciencias Aeroespaciales[2], por el préstamo y colaboración en el uso de los recursos de visualización extendida, para el desarrollo de pruebas.

Al Director del proyecto, Carlos Jaime Barrios Hernandez Ph.D por la ayuda y asesoría durante el desarrollo del proyecto, el tiempo dedicado al seguimiento de los avances y su constante y valiosa colaboración.

Al Codirector del proyecto, Hernan Asorey Ph.D por su apoyo y acompañamiento durante el desarrollo del proyecto, sus enseñanzas y conocimientos compartidos, en el campo de la astrofísica y astronomía.

Los integrantes del grupo de supercomputación y calculo científico de la Universidad Industrial de Santander (SC3UIS). Por sus asesorías y apoyo constante.

[1] http://grid.uis.edu.co [En linea] [Consultado el 13 de febrero del 2015] y http://www.sc3.uis.edu.co [En linea] [Consultado el 13 de febrero del 2015]

[2] Grupo Halley de Astronomía y Ciencias Aeroespaciales http://halley.uis.edu.co [En linea] [Consultado el 13 de febrero del 2015]

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

ACRÓNIMOS

CORSIKA Simulación de Rayos Cósmicos para Kascade (COsmic Ray SImulations for KAscade).

GeV Giga electro-voltios.

GPU : Unidad de Procesamiento gráfico *(Graphic Processing Unit)*.

HPC Computación de Alto Rendimiento *(High Performance Computing)*.

LAGO Large Aperture Gamma Ray Burst Observatory.

SC3 Unidad de Supercomputación y Calculo Científico (UIS).

TeV Tera electro-voltios.

GLOSARIO

Arquitectura Diseño conceptual y estructura operacional de un sistema. Describe un sistema especificando sus partes e relaciones.

Cluster Es un conjunto de computadores conectados para trabajar de manera conjunta.

Concurrencia Es la simultaneidad en la ejecución de múltiples procesos o hilos de ejecución.

Escalabilidad Es la habilidad de un sistema, red o proceso de manejar el crecimiento de su cantidad de trabajo de manera fluida.

Electro-Voltio Es una unidad de energía igual a $1.6e^{-19} joules$ aproximadamente. Es la cantidad de energía ganada (o perdida) por la carga de un electro movido a través de un diferencial de potencia de un voltio.

Granularidad El tamaño de las subtareas en que una tarea puede ser dividida, en relación al aumento de comunicación necesario entre los diferentes hilos para su sincronización.

Framework Es una abstracción en la cual el software provee una funcionalidad general que puede ser cambiada selectivamente por código adicional del usuario.

Librería Una colección de subprogramas usados para desarrollar software, organizados de tal manera que puede ser usado por múltiples programas que no tengan conexión entre si.

Nodo Dispositivos de computo conectados a una red, en Supercomputación se entiende como nodos a los diferentes computadores que hacen parte de un cluster.

Plugin Un componente de software que añade una característica especifica a una aplicación existente.

Programación en Paralelo La computación paralela es una forma de cómputo en la que muchas instrucciones se ejecutan simultáneamente. [7]

Visualización Extendida Observación de imágenes de forma no convencional, permitiendo recorrerla y obtener diferentes niveles de detalle.

RESUMEN

TITULO: VISUALIZACIÓN DE CASCADAS DE RAYOS CÓSMICOS SOBRE GPUS[*]

AUTOR: RAFAEL ANTONIO LAVERDE LAVERDE [**]

PALABRAS CLAVE: Visualización, Computación de Alto Desempeño (HPC) GPUs, VisIt, Astronomía, Rayos Cósmicos

DESCRIPCIÓN:

Para comprender la naturaleza los científicos utilizan simulaciones que debido a su grado de complejidad generan un volumen extenso de datos; los cuales no son suficientes para comprender cabalmente los fenómenos y es necesario transformar estos datos científicos y abstractos en visualizaciones para confrontarlos a la realidad representada.

En astronomía y astrofísica existen muchos problemas que requieren de esa visualización, como es el caso de la simulación de cascadas de rayos cósmicos; La interacción de un rayo cósmico de alta energía ($E > 10GeV$) con la atmósfera produce un fenómeno denominado lluvia atmosférica extendida (EAS, por sus siglas en inglés).Para los rayos cósmicos de mayor energía observados ($E > 10^{11}Gev$), estas lluvias pueden llegar a contener hasta 10^{12} partículas en el punto de máximo desarrollo. Seguir la trayectoria y las posibles interacciones o decaimientos de las partículas en la cascada demandan enormes recursos computacionales, dado que implica: alta complejidad, gran volumen de datos a tratar y estrategias eficientes de visualización para la comprensión de fenómeno.

Actualmente el grupo Halley[3] de la Universidad Industrial de Santander, junto con el proyecto internacional LAGO[4] realizan investigación en astrofísica Solar y astrofísica de rayos cósmicos, apoyándose de simulaciones para la comprobación de datos experimentales.

En el presente trabajo de investigación se analizaron e implementaron diferentes alternativas para visualizar conjuntos extensos de datos correspondientes a cascadas de rayos cósmicos generadas por el software de simulación CORSIKA, utilizando recursos de computo de alto desempeño, computación distribuida y arquitectura cliente-servidor.

Así mismo se analizo la interacción de dicha visualización con plataformas de visualización extendida; y se configuraron los componentes necesarios para lograr dicha interacción.

La implementación de las diferentes estrategias, así como el estudio de la segmentación de los datos, y las pruebas realizadas, permitieron generar lineamientos de las

[*] Trabajo de Grado en la Modalidad de Investigación

[**] Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas e Informática.Director: Carlos Jaime Barrios Hernández. Ph.D. Codirector Hernán Asorey. Ph.D.

[3] Grupo Halley de Astronomía y Ciencias Aeroespaciales http://halley.uis.edu.co [En linea] [Consultado el 13 de febrero del 2015]

[4] Large Aperture Gamma Ray Burst Observatory http://labdpr.cab.cnea.gov.ar/lago [En linea] [Consultado el 13 de febrero del 2015]

estrategias a utilizar para la visualización de conjuntos extensos de datos explotando el paralelismo.

ABSTRACT

Title: Visualization of Cosmic Rays Cascades on gpus[*]

Author: Rafael Antonio Laverde Laverde [**]

Keywords: Visualization, gpus, HPC, VisIt, Astronomy, Cosmic Rays

DESCRIPTION:

For understand the nature scientists use simulations that due its complexity grade generate an extended volume or data; this data is not enough for fully understand the phenomenon and it's necessary to transform this scientific data into visualizations in order to confront with the reality represented.

In astronomy and astrophysics there are many problems that require this visualization, such as cosmic ray air showers simulation; the interaction of a high energy ray ($E > 10GeV$) with the atmosphere produce a phenomenon called Extended Air Shower (EAS). For cosmic rays of higher energy observed ($E > 10^{11}Gev$), this air showers can contain up to 10^{12} particles in the point of highest development. Trace the trajectory and the possible interactions or decay of the particles in the air shower demand huge computing resources, because it involves: high complexity, large volume of data to be processed, and efficient strategies of visualization for understand the phenomenon.

Currently the research group Halley[5] of the *Universidad Industrial de Santander*, together with the international project lago[6] conduct research in Solar and cosmic ray astrophysics, relying on simulations for verify experimental data.

In the present research work were analyzed and implemented different approaches for visualize extensive datasets from cosmic rays cascades generated by the simulation software corsika, using high performance computing resources, distributed computing and client-server architecture.

Furthermore was analyzed the interaction of that visualization with extended visualization platforms, and were set up the components required for accomplish that interaction.

The implementation of the different approaches, as well as the study of the data segmentation, and the tests performed, allow to generate guidelines about the approaches to use for visualization of extensive datasets taking advantage of the parallelism.

---

[*] Undergraduate final project, research modality

[**] Systems Engineering and Computer Science School, Director: Carlos Jaime Barrios Hernández. Ph.D. Codirector Hernán Asorey. Ph.D.

[5] Halley, Astronomy and Aerospace Sciences Group http://halley.uis.edu.co [On line] [Retrieved February 13th 2015]

[6] Large Aperture Gamma Ray Burst Observatory http://labdpr.cab.cnea.gov.ar/lago [On line][Retrieved February 13th 2015]

INTRODUCCIÓN

Actualmente los recursos de computo disponibles y el avance tecnológico han generado un incremento significativo en la cantidad de datos tanto experimentales como sintéticos obtenidos por diversos proyectos en las diversas ramas de la investigación científica.

Este aumento del volumen de datos implica una mejora en las herramientas de análisis para lograr estudios más detallados que aprovechen toda la información contenida en dichos datos. Además los recursos computacionales disponibles actualmente permiten considerar herramientas que anteriormente no eran viables como la visualización, debido a la cantidad de recursos necesarios.

La Unidad de Supercomputación y Calculo Científico UIS (SC3)[7] se ha caracterizado por soportar actividades científicas desarrollando tecnologías de alto rendimiento, y recientemente ha impulsado la implementación de herramientas de visualización de conjuntos extensos de datos, apoyando de esta forma a los proyectos que usan la infraestructura de computo *"GRIDUIS2"* a lograr un análisis más completo de los resultados obtenidos.

El grupo Halley[8] de la Universidad Industrial de Santander, junto con el proyecto internacional LAGO[9] realizan investigación en astrofísica Solar y astrofísica de rayos cósmicos, apoyándose de simulaciones para la comprobación de datos experimentales, para ello hacen uso de los recursos computacionales de la infraestructura *"GRIDUIS2"*, y realizan diversos análisis estadísticos de los datos obtenidos.

La motivación de este proyecto fue las posibilidades y mejoras en el análisis de los datos del proyecto LAGO, que se obtendrían al poder visualizar el desarrollo de las Cascadas de rayos cósmicos con un mayor nivel de detalle; para esto se planteo el uso de arquitecturas de alto desempeño para lograr dicha visualización.

El presente trabajo presenta las diferentes etapas que se siguieron, las alternativas que se consideraron, la implementación de la visualización, su interacción con la plataforma de alto rendimiento *"GRIDUIS2"* y con la plataforma de visualización extendida disponible en el grupo Halley, asó como las pruebas realizadas y los resultados obtenidos. También plantea diferentes lineamientos en la elección de una estrategia de visualización masiva de datos, y en el desarrollo necesario para lograrla.

En las etapas que se siguieron, se considero el análisis al software de simulación CORSIKA, y a la estructura de los datos; y diferentes implementaciones realizadas en varias de las estrategias.

[7]Unidad de Supercomputación y Calculo Científico (Universidad Industrial de Santander) http://sc3.uis.edu.co [En linea][Consultado el 13 de febrero del 2015]

[8]Grupo Halley de Astronomía y Ciencias Aeroespaciales http://halley.uis.edu.co [En linea][Consultado el 13 de febrero del 2015]

[9]Large Aperture Gamma Ray Burst Observatory http://labdpr.cab.cnea.gov.ar/lago [En linea][Consultado el 13 de febrero del 2015]

1 OBJETIVOS

1.1 OBJETIVO GENERAL

Visualizar la simulación de cascadas de rayos cósmicos aprovechando las características arquitecturales de infraestructuras híbridas de cómputo basadas en CPU-GPU.

1.2 OBJETIVOS ESPECÍFICOS

1. Observar la arquitectura de una aplicación existente para la simulación de cascadas de rayos cósmicos, identificando los componentes, procesos e interacciones dirigidos hacia la visualización.

2. Estudiar y elegir la plataforma y la forma en la cual se implementará la visualización de los resultados de la simulación de las cascadas de rayos cósmicos, que soporte la explotación de concurrencia.

3. Seleccionar una estrategia de acoplamiento entre la aplicación y la infraestructura que soporte el software de simulación y la visualización e implementar un prototipo que permita la validación de la estrategia seleccionada.

4. Crear y validar los componentes necesarios para la interacción de la aplicación con plataformas de visualización extendida.

5. Generar lineamientos que permitan la modificación y/o creación de aplicaciones que utilicen visualización de conjuntos extensos de datos explotando el paralelismo.

2 MARCO DE REFERENCIA

2.1 VISUALIZACIÓN PARALELA DE DATOS

Dificultades de la visualización de conjuntos extensos de Datos Visualizar conjuntos extensos de datos es difícil por muchas razones, los códigos actuales de análisis producen conjuntos de datos del orden de Terabytes[10] distribuidos alrededor de miles de nodos de procesamiento. En algunos casos, muchos intervalos de la simulación nunca son guardados en disco, pero deben ser visualizados mientras están en la memoria en los nodos de procesamiento. Esto crea un problema, porque la visualización debe compartir recursos limitados que la simulación esta usando. Este problema se agrava porque la visualización puede requerir más espacio de almacenamiento que la simulación. Otra dificultad es que algunos algoritmos como generación de línea de corriente o destrucción de malla, no están diseñados para operar con datos distribuidos o en paralelo. [1]

Técnicas de visualización científica en Astronomía Las técnicas más comunes para presentar datos escalares tridimensionales en astronomía son: puntos, splats[11], iso-superficies (superficies de valor constante), y *renderizado* de volúmenes (figura 1). [8]

Estructuras de manejo de datos para explotar paralelismo en problemas de visualización extensiva de datos Una forma de clasificar los problemas de visualización es basándose en cómo los datos son representados mediante programación, es decir cómo son guardados en la memoria y/o en el disco: [8]

Puntos dispersos Los datos están compuestos de un conjunto de puntos (x, y, z) y tienen asociados datos de atributos (ej: densidad, presión, y temperatura) (figura 2, arriba izquierda).

Grid Estructurado Los valores de los datos son especificados en un grid regular de 3 dimensiones, con sus celdas alineadas con los ejes cartesianos (figura 2, arriba derecha).

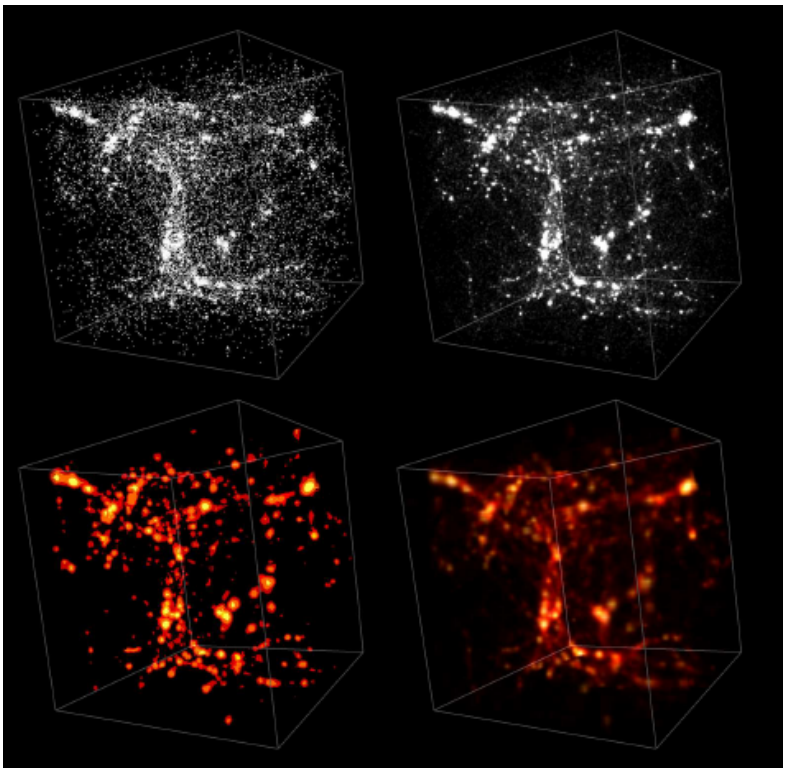
Grid no estructurado Los valores de los datos son especificados en las esquinas de un elemento de 2 o 3 dimensiones con una conectividad explicita definida (figura 2, abajo izquierda).

Grid Adaptativo Los valores de los datos son especificados en un grid estructurado de varias resoluciones. Un grid grueso es usado para cubrir todo el dominio computacional, combinado con sub-grids superpuestos para lograr una mayor resolución para regiones de interés, por ejemplo donde la densidad de partículas es mayor (figura 2, abajo derecha).

[10] Unidad de memoria o almacenamiento de datos equivalente a 10^{12} bytes.

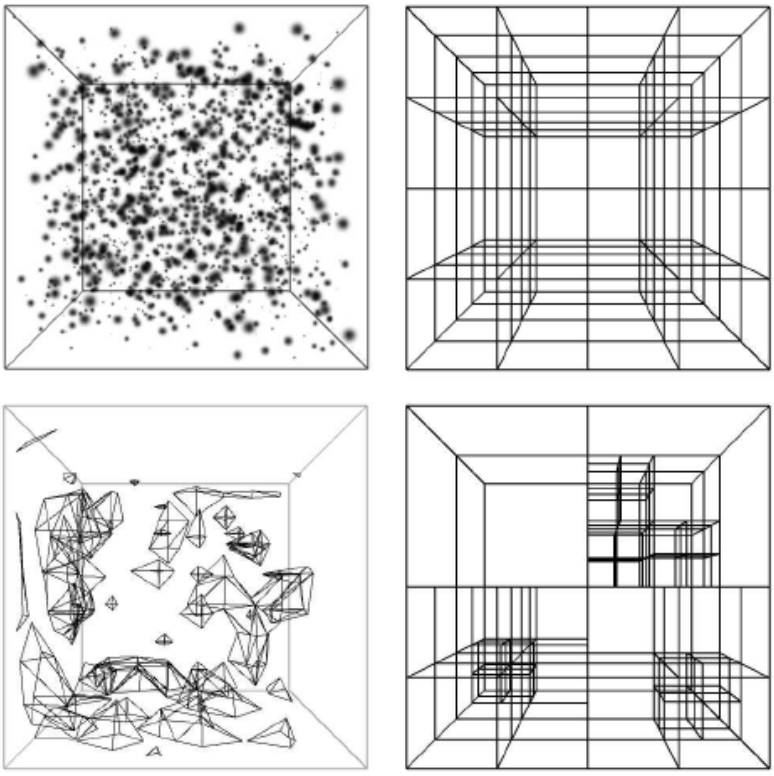
[11] Técnica de *renderizado* en la que cada elemento del volumen es aplastado como una bola de nieve.

Figura 1: Cuatro formas comunes de visualización aplicadas a un conjunto de datos cosmológico de N-Cuerpos



Fuente: Scientific Visualization in Astronomy: Towards the Petascale Astronomy Era[8]

Figura 2: Las cuatro representaciones estándar usadas en computación científica. Puntos de datos dispersos (arriba izquierda); grid estructurado (arriba derecha); grid no estructurado (abajo izquierda); y grid adaptativo (abajo derecha).



Fuente: Scientific Visualization in Astronomy: Towards the Petascale Astronomy Era[8]

Plataformas para visualización de datos explotando paralelismo

VTK Es un sistema de software libre y de código abierto para computación gráfica en 3D, procesamiento de imágenes y visualización. VTK consiste en un librería de clases de C++ y varias capas de interfaz interpretadas incluyendo Tcl, Java y Python.

VTK soporta una amplia variedad de algoritmos de visualización: escalar, vector, tensor, textura, y métodos volumétricos; y técnicas de modelamiento avanzado tales como: modelado implícito, reducción poligonal, malla de suavizado, corte, contorneado y triangulación Delaunay[12].

VTK es multiplataforma y funciona en Linux, Windows, Mac y plataformas Unix. [14]

Paraview Es una aplicación de código abierto y multiplataforma para el análisis y visualización de datos. Permite constituir rápidamente visualizaciones y analizar datos usando técnicas cuantitativas y cualitativas. Paraview fue desarrollado para analizar conjuntos de datos extremadamente largos del rango de Terabytes usando recursos distribuidos de memoria y cómputo. Este puede funcionar en supercomputadores para analizar conjuntos de datos del rango de los Terabytes así como en computadores personales para conjuntos de datos pequeños. [13]

OpenGL Es el primer entorno de desarrollo portable e interactivo para aplicaciones gráficas de 2 y 3 dimensiones. Cualquier aplicación que requiera el máximo desempeño en animaciones 3D o simulaciones visuales puede explotar las capacidades de OpenGL, es multiplataforma y su API[13] es compatible con los lenguajes de programación más populares. [19]

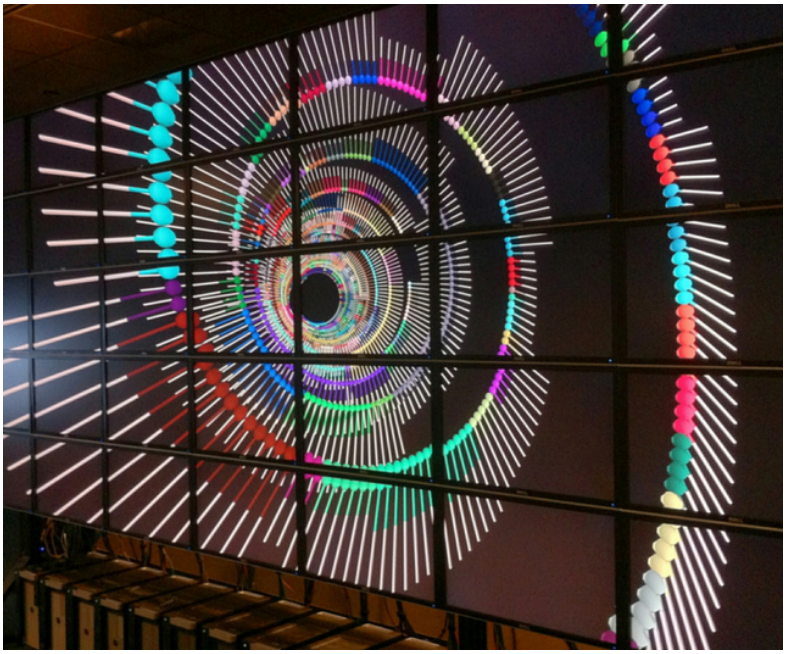
VisIt Es una herramienta de visualización libre, de código abierto, multiplataforma, distribuida y paralela, para visualizar datos definidos en mallas de dos o tres dimensiones estructuradas o no-estructuradas. La arquitectura distribuida de VisIt le permite tomar ventaja tanto de un gran computador paralelo como del hardware de aceleración gráfica de un computador local. La interfaz gráfica es frecuentemente ejecutada localmente, mientras el componente del motor de computo se ejecuta en paralelo en un computador remoto. La arquitectura distribuida de VisIt permite visualizar los datos donde fueron generados, eliminando la necesidad de mover los datos a un servidor de visualización. VisIt puede ser controlado desde la Interfaz Gráfica de Usuario (GUI), y a través de los lenguajes de programación Python y Java. o desde una interfaz personalizada que se desarrolle.[16]

[12]Consiste en una red de triángulos que cumple la condición de Delaunay, es decir que la circunferencia circunscrita de cada triángulo de la red no debe contener ningún vértice de otro triángulo.

[13]Conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software.

Interacción con plataformas de visualización extendidas La interacción con plataformas de visualización extendida como las que actualmente se dispone en el grupo SC3[14] se realiza mediante el uso de un paquete de código abierto llamado *DisplayCluster*[15] desarrollado por el TACC[16]. Este paquete de software es multiplaforma, permite la reproducción de contenidos de vídeo, píxel streaming[17], y es programable mediante *scripts* en Python. [12] La figura 3 presenta una pantalla dividida ejecutando *DisplayCluster*.

Figura 3: Cluster de visualización extendida del TACC, ejecutando DisplayCluter un ambiente de software para manejar interactivamente pantallas dividas de gran escala.



Fuente: Pagina web del TACC[12]

2.2 GENERALIDADES SOBRE ASTRONOMÍA DE RAYOS CÓSMICOS

Definición Los Rayos Cósmicos son partículas de alta energía originadas fuera del Sistema Solar. [20] Estas pueden producir lluvias de partículas secundarias al penetrar la atmósfera terrestre, las cuales algunas veces alcanzan la superficie.(figura 4)

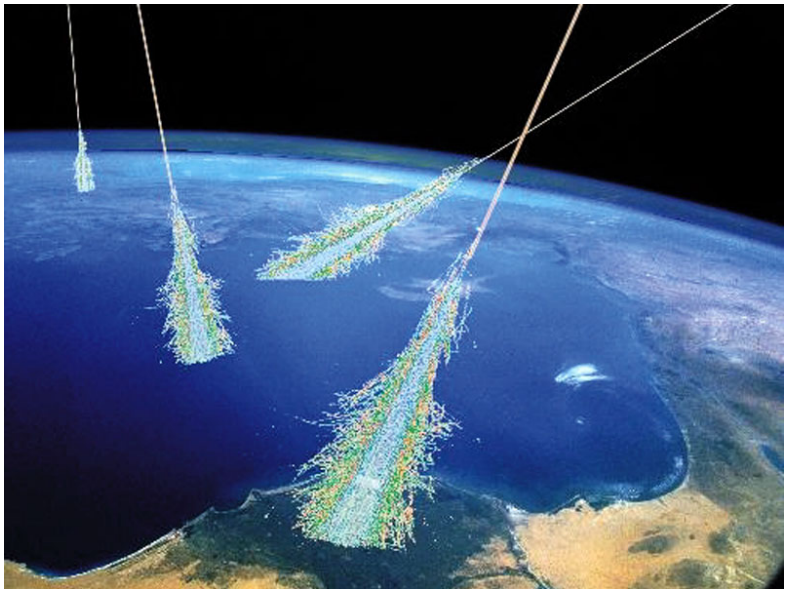
[14]Unidad de Supercomputación y Calculo Científico http://sc3.uis.edu.co [En linea][Consultado el 13 de febrero del 2015]

[15]http://www.tacc.utexas.edu/tacc-software/displaycluster [En linea][Consultado el 13 de febrero del 2015]

[16]Centro de Cómputo avanzado de Texas, http://www.tacc.utexas.edu/ [En linea][Consultado el 13 de febrero del 2015]

[17]Reproducir un flujo de vídeo proveniente de un entorno de escritorio o aplicación

Figura 4: Imagen ilustrativa de una lluvia de rayos cósmicos originada al entrar a la atmósfera.

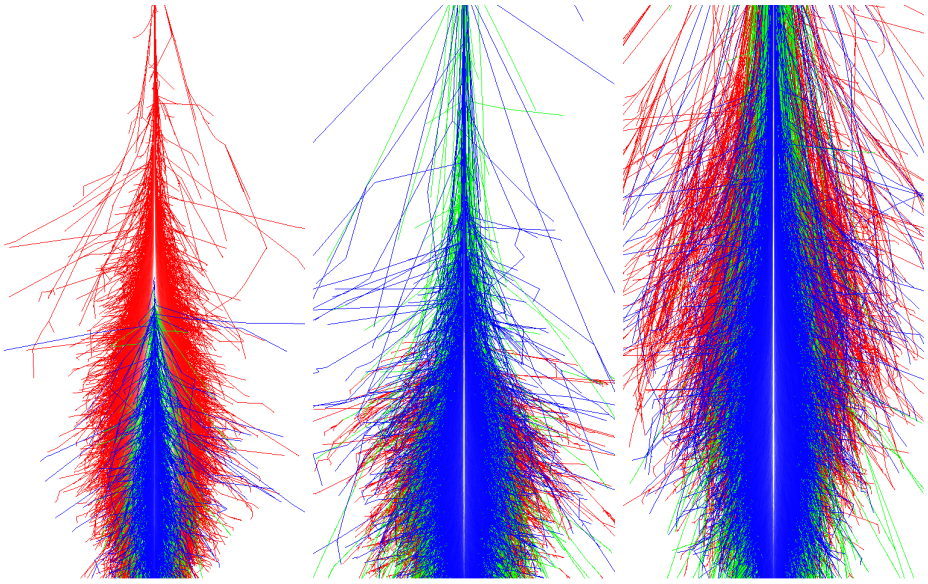


Fuente: Pagina web NASA[21]

Cascadas de Rayos cósmicos La interacción de un rayo cósmico (primario) con la atmósfera, lo cual ocurre típicamente a alturas de entre 15 km y 40 km sobre la superficie terrestre, genera una cascada de partículas (secundarios) cuya cantidad es proporcional a la energía del primario. A este fenómeno se conoce en la literatura como Lluvias Atmosféricas Extendidas (EAS, por sus siglas en inglés, *Extensive Air Showers*). Las partículas de la cascada se distribuyen en tres componentes principales: la componente electromagnética (EM, formada por electrones $e^{\pm}$ y fotones γ), la componente muónica (formada por muones $\mu^{\pm}$ provenientes del decaimiento de mesones cargados, típicamente $\pi^{\pm}$ y $K^{\pm}$), y la componente hadrónica (formada por hadrones, nucleones y algunos núcleos livianos). La relación numérica entre las tres componentes está directamente relacionada con el tipo de primario, pero siempre se da que la componente electromagnética domina la distribución numérica ($\sim 70\,\%$) y la distribución en energía ($\sim 85\,\%$ de la energía del primario se encuentra en esta componente). Como ejemplo, en la figura 5 puede verse una simulación de la evolución de tres EAS originadas por un fotón, un protón y un núcleo de hierro, todas con una energía $E = 5 \times 10^{14}\,\text{eV}$, ingresando a la atmósfera de manera vertical sobre el detector. Resulta claro ver la evolución de las tres componentes individuales, y como estas varían en su distribución en función del tipo de rayo cósmico que ingresó en la atmósfera. [9]

La evolución de la cascada resulta entonces de la composición de dos factores claves: la producción de nuevas partículas por reacciones nucleares y decaimientos de partículas inestables, y la absorción de partículas de baja energía en la atmósfera.

Figura 5: Desarrollo longitudinal y lateral de tres lluvias atmosféricas extendidas iniciadas por un fotón (izquierda), un protón (centro) y un hierro (derecha), todos verticales y con $E_p = 5 \times 10^{14}$ eV. Los diferentes colores identifican las tres cascadas principales: electromagnética (rojo), muónica (verde) y hadrónica (azul). Las diferencias son evidentes: mientras que la componente EM domina la lluvia iniciada por un fotón, la hadrónica se hace más importante al aumentar el número de nucleones.



Fuente: Los detectores Cherenkov del Observatorio Pierre Auger y su aplicación para el estudio de fondos de radiación [2]

2.3 CORSIKA SIMULADOR DE CASCADAS DE RAYOS CÓSMICOS

CORSIKA[18] es un programa para simulaciones detalladas de cascadas extensas iniciadas por partículas de rayos cósmicos de altas energías. Protones núcleos livianos hasta el hierro, fotones, y muchas otras partículas pueden ser consideradas como primarias. Las partículas son seguidas a través de la atmósfera mientras experimentan reacciones o - en caso de secundarias inestables - decaen. [9]

El código permite simular las cascadas producidas por numerosos tipos de primarios (protones, núcleos, electrones, fotones, etc.), y ha sido ampliamente utilizado para validar las observaciones en numerosos experimentos y observatorios en un amplio rango de energías. [4]

La amplia aceptación del código y sus resultados, así como su continuo desarrollo y mantenimiento, hacen que esta sea la opción más confiable a la hora de realizar la simulaciones necesarias para LAGO. [4]

[18] COsmic Ray SImulations for KAscade http://www-ik.fzk.de/~corsika [En linea][Consultado el 13 de febrero del 2015]

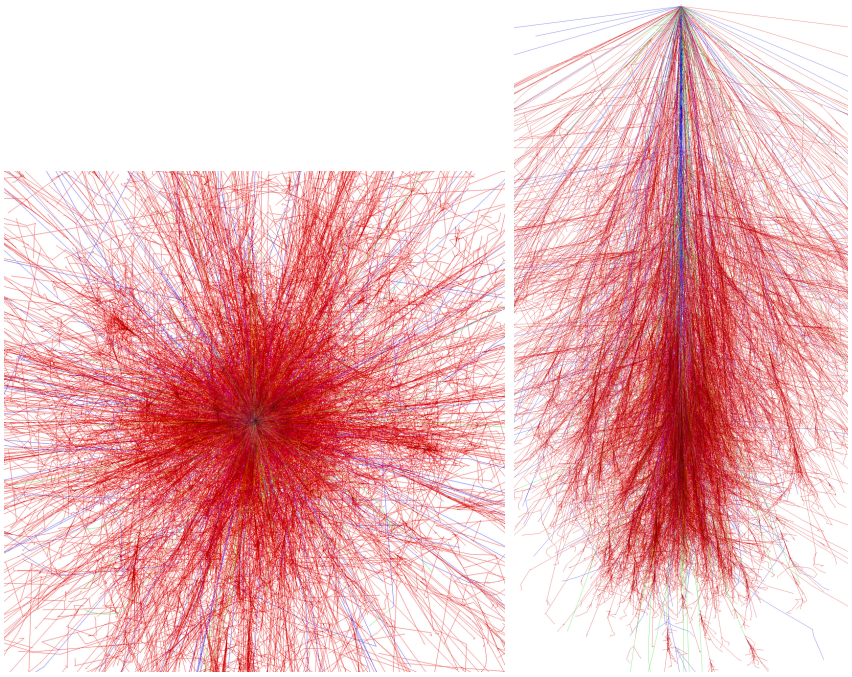
3 ESTADO DEL ARTE

3.1 ESTADO ACTUAL DE LA VISUALIZACIÓN EN CORSIKA

Junto con los archivos de CORSIKA vienen varias utilidades para el manejo de los resultados de la simulación, entre estas una herramienta para generar gráficos en formato *png* la cual permite observar la dispersión de la lluvia y mediante colores diferenciar las partículas primarias y secundarias (figura 6), pero no es útil sí se desea hacer un análisis más a fondo de los datos, y lograr deducir información de mayor utilidad.

Esta utilidad permite visualizar imágenes de dos dimensiones desde diferentes perspectivas, y da una noción general del fenómeno, pero no permite ningún nivel de detalle, ni la diferenciación especifica de cada partícula, y tampoco representa claramente la energía de cada partícula.

Figura 6: Diferentes proyecciones del desarrollo de una lluvia atmosférica extendida iniciada por un protón de hierro(56) con una energía de ($E = 10^{12} eV$ (TeV)), proyección xy (izquierda) y proyección xz (derecha) (Imágenes tomadas de la galería disponible en la pagina web de CORSIKA).



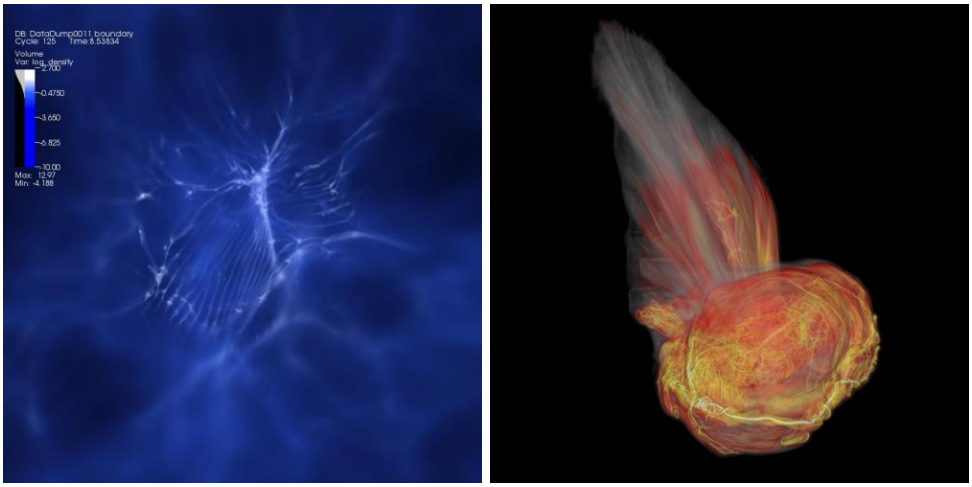
Fuente: Galería de la pagina web de CORSIKA[18]

3.2 ESTADO ACTUAL DE LA VISUALIZACIÓN CIENTÍFICA DE CONJUNTOS EXTENSOS DE DATOS

Gracias a las herramientas como VisIt y Paraview, se ha logrado la visualización de datos experimentales o sintéticos, permitiendo una representación más clara del fenómeno y un mejor análisis. Por ejemplo la figura 7 representa una formación estelar y fue creada a partir de gráficar el logaritmo de la densidad del gas y/o polvo estelar; y la figura 8 representa una supernova, mediante renderizado de volumen de la magnitud del vortice, una medida de turbulencia, lo cual ayuda a identificar una *"burbuja"* dentro de la supernova.

Figura 7: Formación estelar (izquierda): Es una gráfica del volumen del logaritmo del gas/polvo estelar en la estrella Enzo . Regiones de alta densidad son blancas mientras las menos densas son más azules y más transparentes.

Figura 8: Supernova (derecha): La gráfica representa un renderizado de volumen de la magnitud del vortice, una medida de turbulencia; para generar esta imagen VisIt fue usado en el supercomputador *NSCA Blue Waters* para visualizar la salida de CASTRO, un código masivamente paralelo, de volumen finito, con refinamiento adaptativo de malla (AMR). (Imagen creada por el staff de visualización del Blue Waters).

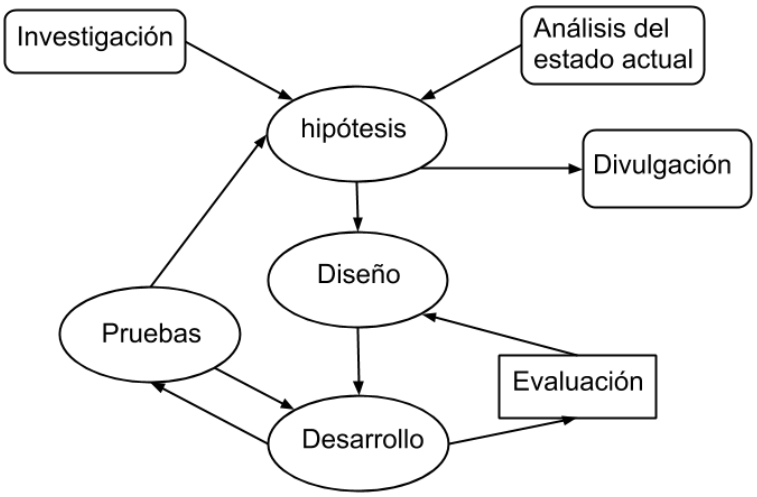


Fuente: Galería de la pagina web de VisIt [17]

4 METODOLOGÍA

El proceso metodológico que se siguió fue el método científico, a través de este se ve el análisis, diseño, implementación y validación del componente de software para la visualización de los rayos cósmicos. Las etapas se organizaron como en la figura 9.

Figura 9: Diagrama de la metodología a usar, corresponde a una adaptación del método científico.



El proyecto se realizo en 4 fases que se ejecutaron en constante realimentación.

4.1 FASE 1: DOCUMENTACIÓN Y BOSQUEJO DEL DISEÑO DE LA SOLUCIÓN

Se inició el proceso de documentación e investigación en el área de astrofísica de rayos cósmicos, con el fin de definir la forma en que se visualizarían las cascadas para que fueran de mayor utilidad en la investigación; al mismo tiempo se realizó un proceso análogo en el área de visualización, para definir la plataforma y/o librerías que se usarían en el desarrollo.

Se realizaron bosquejos en papel de la forma en que podría ser el flujo de datos desde salida de datos del simulador, y de la forma en el que se paralelizara el cómputo de los datos.

Así mismo se hicieron varias implementaciones de prueba en distintas plataformas, para analizar la viabilidad de cada una, y escoger la más adecuada.

4.2 FASE 2: DESARROLLO DEL *BACK-END* Y PRIMER PROTOTIPO DE LA VISUALIZACIÓN

Se desarrolló el componente encargado del manejo del *stream* de datos y de la forma en como se distribuirían sobre los recursos disponibles, y se implementó un primer prototipo

de la visualización en el *framework* elegido anteriormente, se realizaron pruebas y se definió la mejor forma de subdividir los datos para la paralelización del cómputo.

4.3 FASE 3: CONFIGURACIÓN DEL FRONTEND Y DE LA ARQUITECTURA CLIENTE SERVIDOR

Luego de desarrollar el componente encargado de la lectura y de subdividir los datos se procedió a definir y realizar la configuración por defecto del frontend de la herramienta de acuerdo a las características definidas en la primera fase, se visualizaron distintas simulaciones de lluvias de cascadas de rayos cósmicos, para realizar pruebas de desempeño y consumo de recursos, realimentar y mejorar el componente de visualización.

Se configuró la interacción de la arquitectura cliente-servidor con el fin de aprovechar recursos remotos de un supercomputador para acelerar el computo y lograr un mayor manejo de datos.

Se realizó la integración para lograr un funcionamiento óptimo de la aplicación sobre la plataforma de visualización extendida. Se analizó el comportamiento de la aplicación de acuerdo a los acciones ejecutadas por un usuario externo. Se definió los lineamientos en los cuales la aplicación se comportará con éxito, la escalabilidad y los recursos de hardware necesarios para un funcionamiento fluido.

4.4 FASE 4: VALIDACIÓN Y EXPERIMENTACIÓN

En esta última fase se probaron diferentes visualizaciones y se evaluó su utilidad en el análisis de astrofísica de rayos cósmicos, se realimentó y se hicieron mejoras para lograr resultados más útiles.

5 ANÁLISIS DE LA ARQUITECTURA DE CORSIKA

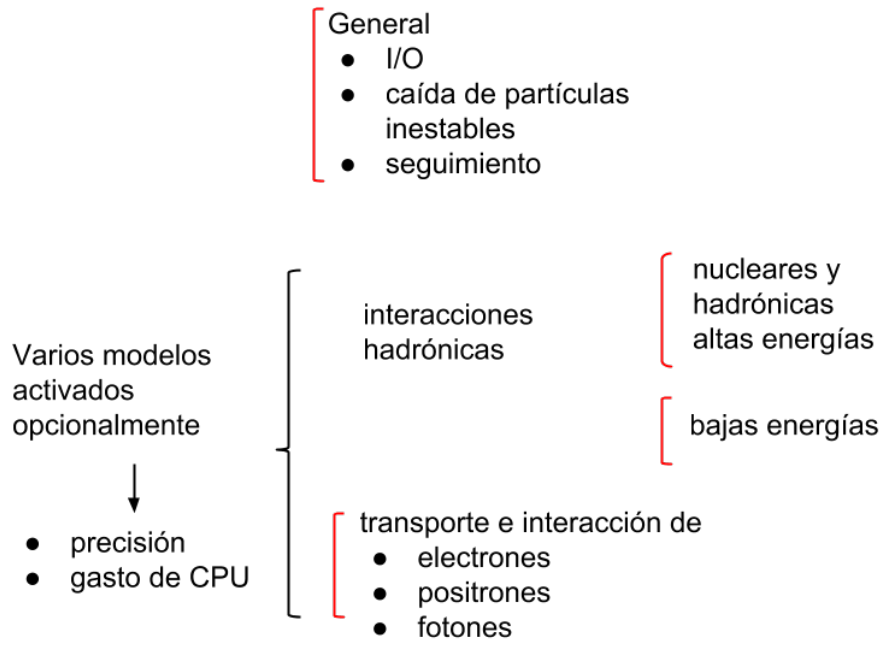
5.1 CARACTERÍSTICAS TÉCNICAS DE CORSIKA

CORSIKA es un conjunto completo de rutinas de FORTRAN[19]. No usa programas o librerías adicionales para la simulación de rayos cósmicos; aunque las versiones actuales incluyen librerías auxiliares en C y C++.[10]

5.2 MÓDULOS Y FUNCIONAMIENTO DEL SIMULADOR

Consiste principalmente de 4 partes: La *primera* parte es un programa general que maneja la entrada y salida, realizando el decaimiento de partículas inestables, y rastreando las partículas teniendo en cuenta la pérdida de energía de ionización y deflexión por múltiples dispersiones y el campo magnético de la tierra. La *segunda* parte trata las iteraciones hadrónicas de núcleos y hadrones con el aire nuclear a altas energías. La *tercera* parte simula las interacciones hadrónicas a bajas energías y la *cuarta* parte describe el transporte e interacción de electrones, positrones y fotones. [10] (figura 10)

Figura 10: Diagrama de la estructura de los componentes de CORSIKA de acuerdo a su función en la simulación.



[19] FORTRAN Lenguaje de programación imperativo de propósito general.

5.3 ESTRUCTURA DEL DESARROLLO DE CORSIKA

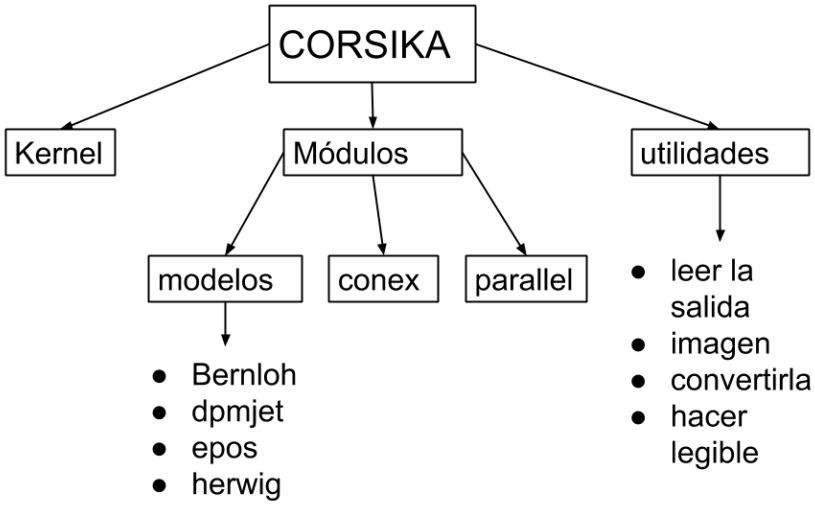
En la estructura del código fuente de CORSIKA, permite apreciar tres clasificaciones, de los archivos. (figura 11)

Kernel Parte central de la simulación, compuesta por varios archivos contenidos en el subdirectorio `/src`, principalmente por los archivos `corsika.F` y `corsika.h`.

Módulos Distintos modelos para simular diferentes tipos de fenómenos de interacción de partículas, que se activan o desactivan en la compilación de CORSIKA, compuesta por varios subdirectorios, cada uno con el nombre del modulo (`bernlohr, coast, conex, dpmjet, etc ...`). Así como algunos *scripts* de bash y rutinas de C, para ejecutar CORSIKA de manera distribuida sobre un cluster de CPUs.

Utilidades Programas no muy extensos ubicados en `/src/utils`, que permiten el análisis de los datos arrojados por el simulador, por ejemplo, leerlos, transformarlos a texto plano, restablecer el archivo de entrada que genero la simulación, dividir una salida muy grande de acuerdo a los eventos, realizar gráficas de las trayectorias, entre otros.

Figura 11: Diagrama de la estructura de los componentes de CORSIKA de acuerdo a la estructura del árbol de directorios, y el tipo de función realizada.



5.4 LIMITACIONES Y CONSIDERACIONES GENERADAS A PARTIR DEL ANÁLISIS

En el desarrollo del simulador se observa algunas falencias, CORSIKA no tiene una arquitectura con una estructura clara, y algunas partes son una colección de rutinas que no tienen una fuerte relación entre si, estas falencias son das a la forma en que se

ha realizado su desarrollo, y que cuando se inicio su desarrollo no existían tecnologías, arquitecturas y metodologías que existen actualmente, y al hecho que diferentes modelos de comportamiento de partículas son desarrollados por otros proyectos.

Del estudio de la arquitectura de la aplicación, se plantea las dificultades para modificar directamente el código del simulador, lo cual limita las posibilidades planteadas para el acoplamiento de la aplicación con la implementación que se realice para la visualización.

Cabe mencionar una consideración extra, no orientada netamente a la visualización: Al comienzo del proyecto se planteo la posibilidad de paralelizar el kernel o algunos módulos del simulador para no solamente mejorar la visualización, sino acelerar el proceso de simulación, pero debido a la naturaleza del problema y su granularidad no era óptimo para ser paralelizado mediante el paradigma SIMD utilizado en las tarjetas gráficas, además de las dificultades por la estructura su arquitectura.

6 PLANTEAMIENTO Y ANÁLISIS DE LAS ESTRATEGIAS DE VISUALIZACIÓN

6.1 ESTRUCTURA DE LOS DATOS GENERADOS POR CORSIKA

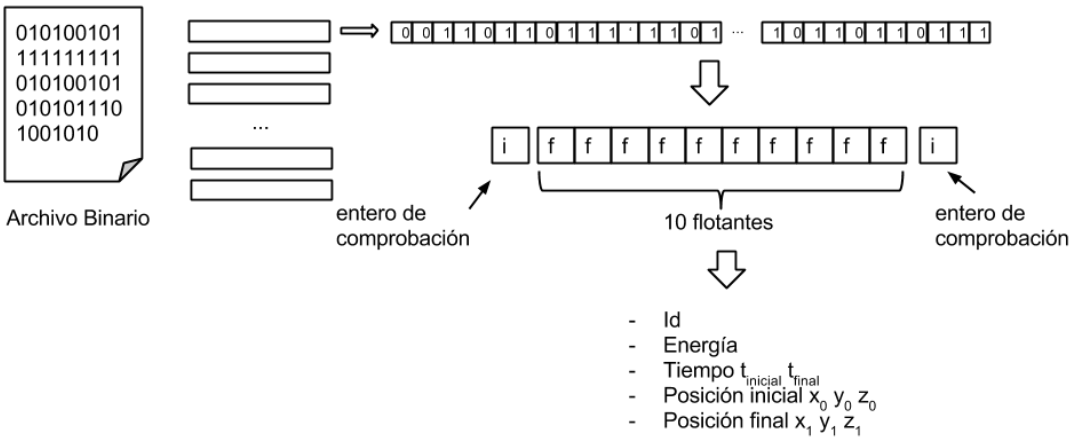
De acuerdo a las opciones de compilación y las de inicio de la simulación, CORSIKA crea diferentes archivos, con los datos generados, para el propósito del presente proyecto son útiles los generados por la opción `PLOTSH`.

Los archivos `PLOTSH` contiene el seguimiento de todos los pasos de cada partícula con el punto inicial y final para producir gráficas que muestren el desarrollo de las lluvias. Estos son escritos dentro del directorio que se indique, el archivo *'DATnnnnn.track_em'* para las particula electro-magneticas, *'DATnnnnn.track_mu'* para los muones, y *'DATnnnnn.track_hd'* para los hadrones, *nnnnnn* es el numero especifico de la ejecución. [10]

Analizando detalladamente el archivo principal `corsika.F` así como algunas de las rutinas usadas para el análisis de los archivos de salida, se pudo determinar la estructura de los datos contenidos en cada uno de los archivos *DATnnnnn.track_{em, mu, hd}*.

Estos archivos son binarios, y contienen secuencialmente estructuras de FORTRAN, conformadas por 10 números de punto flotante representando, el identificador de la partícula, tiempo inicial y final (t_0, t_1), energía, y posición inicial y final $(x_0, y_0, z_0, x_n, y_n, z_n)$; además de estar delimitado por dos enteros al inicio y final con la longitud en bytes de la estructura (40 *bytes*), método de comprobación usado por FORTRAN en la escritura de archivos binarios. (figura 12)

Figura 12: Diagrama de la estructura de los archivos binarios generados por el simulador CORSIKA, contiene entradas de 10 números de punto flotante correspondientes a: El identificador de la partícula, tiempo inicial y final (t_0, t_1), energía, y posición inicial y final $(x_0, y_0, z_0, x_n, y_n, z_n)$.



6.2 LIMITANTES EN LA IMPLEMENTACIÓN

Antes de considerar las estrategias, se analizaron los diferentes factores que podrian influir y limitar el manejo de los datos en la visualización.

Tamaño de los archivos CORSIKA solamente genera tres archivos conteniendo la trayectoria (uno por cada tipo general de partícula), por lo cual el tamaño de estos suele ser considerable, alcanzando tamaños del orden de los $100GBs$ para simulaciones de rayos cósmicos de mayor energía observadas ($E > 10^{11}Gev$). La existencia de pocos archivos limita la granularidad del problema si se quiere paralelizar la lectura de los archivos.

Estructura de los datos Los datos son guardados de la forma *array de estructuras* (debido a que en este orden se generan), lo cual hace ineficiente la lectura del archivo dado que para el manejo masivo de datos es mejor la forma *estructura de arrays*; por ejemplo si se quiere leer los datos correspondientes a la posición inicial de las partículas (30 % del archivo aproximadamente), se tendrá que leer todo el archivo desperdiciando el 70 % de la lectura porque los datos están dispersos.

Formato no estándar La forma en que son guardados los datos, no representa ningún formato estándar conocido.

6.3 ESTRATEGIAS PLANTEADAS PARA LA VISUALIZACIÓN

De acuerdo a las limitantes planteadas anteriormente y al estudio realizado de las diferentes plataformas, librerías, o frameworks disponibles para visualización se consideraron las siguientes estrategias, algunas para partes especificas del proceso, que se pueden combinar entre si generando más opciones (figura 13):

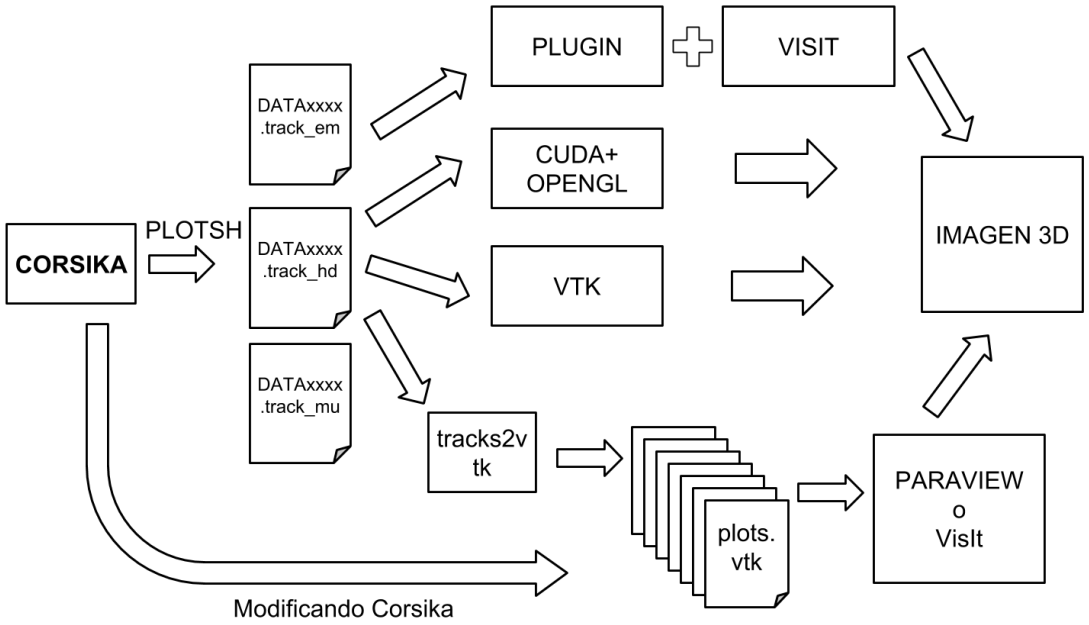
Convertir el archivo de salido a un formato estándar (vtk o silo) El cambiar el formato del archivo permitiría un mejor manejo posterior dato que seria mas óptima la lectura, además de ser más paralelizable. Y podría ser directamente por una herramienta de Visualización como Paraview o VisIt.

Modificar el código fuente de CORSIKA para generar archivos estándar Esta estrategia ahorraría el tiempo y espacio en disco extra utilizado por la conversión.

Desarrollar un plugin en C++ para la herramienta de visualización VisIt También evitaría el tener que realizar la conversión, y los datos podrían ser leídos directamente por la herramienta de visualización.

Realizar un desarrollo en C++ con librerías de VTK Permitiría un mayor control del manejo de los datos, y de la visualización.

Figura 13: Diagrama de las estrategias consideradas para el flujo de datos entre el simulador y la forma de visualización.



Realizar un desarrollo en C++ usando OpenGL y CUDA Permitiría un control aún mayor, no solamente sobre los datos, sino a bajo nivel, sobre el hardware.

Instrumentar el código de la simulación El software de visualización VisIt provee una librería de tamaño modesto que contiene diferentes funciones, para instrumentar el código, cuando el código de simulación es instrumentado, VisIt puede conectarse a este y acceder a los arrays que este exponga.

6.4 VENTAJAS Y DESVENTAJAS DE LAS ESTRATEGIAS

Los parámetros mas relevantes considerados para elegir la estrategia a implementar, fueron: El espacio extra necesario en Disco Duro para alojar datos convertidos, el control del flujo de datos, y la viabilidad, esta última reflejada en la dificultad de la estrategia, y en la cantidad de tiempo de desarrollo necesario; el la tabla 1 se puede apreciar la comparación de dichos parámetros.

Tabla 1: Comparación de las estrategias consideradas para la visualización, en combinaciones de estrategias hay dos valores cada uno correspondiente a una parte.

Estrategia	Espacio extra HD	Control sobre el manejo de datos	Dificultad	Desarrollo necesario
Convertir + Paraview o VisIt	SI	Limitado	Bajo, Bajo	Poco
Modificar CORSIKA + Paraview o VisIt	NO	Limitado	Alta, Bajo	Medio
Convertir + Desarrollo en VTK	SI	SI	Bajo, Alto	Poco, Mucho
Modificar + Desarrollo en VTK	NO	SI	Alto, Alto	Medio, Mucho
Convertir + Desarrollo en OpenGL + CUDA	SI	SI	Bajo, Alto	Poco, Mucho
Modificar + Desarrollo en OpenGL + CUDA	NO	SI	Bajo, Alto	Poco, Mucho
Plugin para VisIt	NO	Medio	Medio	Medio

Estrategia Seleccionada: Plugin para VisIt De acuerdo a la comparación realizada se seleccionó el realizar un plugin para el software de Visualización VisIt, ya que es una de las estrategias más viables, y además los resultados obtenidos con esta estrategia serian acordes a los planteados.

La estrategia de **convertir el archivo**, fue descartada, por la cantidad de espacio de Disco Duro, el ser necesario un paso mas para lograr la visualización, el tiempo extra

requerido por este paso, y que algunos tipos de archivo no permitían representar todos los valores contenidos en los datos.

La estrategia de **modificar CORSIKA** para la **creación de archivos compatibles** o **instrumentar el código**, fue descartada, dado la complejidad del código, el cual esta desarrollado en FORTRAN, lo cual hubiera requerido un tiempo extra en el dominio del lenguaje de programación y de la estructura del proyecto; además seria necesaria la integración del código, para no tener que volver a incluir el desarrollo en cada versión nueva de CORSIKA.

La estrategia de hacer un **desarrollo con librerías de VTK**, fue descartada, dado que implicaba un desarrollo mucho mas extenso, y los resultados serian muy similares, dado que los software de Visualización considerados (Paraview y VisIt) emplean dichas librerías para el calculo y generación de la visualización.

La estrategia de un **desarrollo en CUDA y OpenGL** fue descartada, dado que el desarrollo era aun mas extenso y complejo, lo cual hubiera implicado obtener unos resultados no muy buenos, o un tiempo de desarrollo mucho mayor. Y las herramientas de visualización VisIt y Paraview están desarrolladas a partir de VTK, el cual internamente usa la librería OpenGL; y estas herramientas también permiten el uso de las tarjetas gráficas para realizar el renderizado, por lo cual esta estrategia no representaría ninguna mejora significativa al resultado que se podría obtener con la estrategia seleccionada.

7 IMPLEMENTACIONES REALIZADAS PARA EL ANÁLISIS Y VISUALIZACIÓN

Durante el análisis de la estructura de los datos, y de las diferentes alternativas de visualización se realizaron algunas implementaciones un poco más simples, además de la estrategia escogida.

Estas implementaciones se encuentran disponibles en el manejador de repositorios privado del grupo SC3 en el proyecto *Viz-corsika*[20], en dos repositorios, uno con la implementación estrategia final, y otro con las implementaciones de experimentación.

7.1 SCRIPT EN PYTHON PARA CONVERTIR LOS DATOS A CSV

El primer análisis fue mediante ingeniería inversa determinar la forma en como eran escritos los archivos *DATnnnnn.track_{em, mu, hd}*, para esto se reviso el archivo principal `corsika.F` y algunas de las rutinas contenidas en `src/utils`, se analizo la estructura de datos de FORTRAN usada al escribir los datos.

El script de Python lee el archivo binario en pedazos *chunks* de 48*bytes*, los cual luego desempaqueta convirtiéndolos a un array conformado por dos enteros y 10 flotantes, los cuales escribe en un archivo csv *(comma-separated values)*.

Para la conversión de datos binarios, utiliza la librería *struct* [21].

Luego de la implementación se procedió con la comprobación de esta mediante la comparación de estos datos con los esperados en este tipo de fenómenos, y la validación de un experto.

7.2 IMPLEMENTACIÓN EN C++ PARA CONVERTIR LOS DATOS A VTK

Programa en C++ para leer los datos contenidos en los archivos de entrada, y usando librerías VTK crea una malla no estructurada conteniendo las trayectorias de las partículas; utiliza una estructura de C conformada por 2 enteros y 10 números de punto flotante para leer los datos.

Esta implementación tenia la limitante de no poder asociar los demás valores contenidos (energía, tiempo, id) en el mismo archivo, y era necesario crear varios archivos guardando los datos correspondientes a los demás valores.

7.3 IMPLEMENTACIÓN C++ PARA CONVERTIR LOS DATOS A SILO

Programa en C++ para leer los datos contenidos en los archivos de entrada y usando librerías que forman parte de VisIt guarda los datos en un archivo de extención *.silo*, este tipo de archivos es el archivo por defecto que usa VisIt, esta implementación permite

[20] http://forge.sc3.uis.edu.co/redmine/projects/viz-corsika [En linea][Consultado el 13 de febrero del 2015]

[21] https://docs.python.org/3.4/library/struct.html [En linea][Consultado el 13 de febrero del 2015]

crear una malla y asociar los demás valores; para la lectura de los datos se siguió el mismo procedimiento que la rutina para convertir a VTK.

En esta rutina se tiene la limitante de no soportar la escritura de mallas no estructuradas y por consiguiente se guarda una nube de puntos, lo cual no representa la trayectoria solamente las posiciones iniciales y finales.

7.4 PLUGIN DE VISIT PARA LEER LOS DATOS DIRECTAMENTE

Conformado por varios archivos de código en C++, un archivo para la compilación del Software (`CMakeLists.txt`)y archivo xml definiendo la estructura del plugin (`corsika_reader.xml`). Esta estrategia permitió leer todos los datos, crear una malla no estructurada para representar las trayectorias, así como una nube de puntos para representar las posiciones iniciales y finales, y asociar a estas mallas los demás valores (energía, identificación de la partícula y tiempo), se hablara con más detalle de esta implementación en la sección 8.

7.5 CONFIGURACIONES PARA MEJORAR LA VISUALIZACIÓN

Para una mejor presentación y análisis de los datos, se crearon varios filtros, y algunos configuraciones de valores para algunos gráficos. Estas configuraciones pueden ser guardadas en archivos del lenguaje de marcado xml, los cuales pueden ser cargados luego cuando sean requeridos.

Entre estas configuraciones hay varios filtros de umbral *(threshold)* para definir limites a algunas variables como la energía y poder observar solamente partículas de un rango de energía. También hay algunas tablas de color para ver las partículas de mayor interés con colores específicos.

7.6 ARQUITECTURA CLIENTE-SERVIDOR

Las visualizaciones científicas casi siempre con ejecutadas en un potente supercomputador y accedidas usando un computador de características normales. VisIt puede correr en un modo distribuido que permite este exacto caso de uso; la interfaz gráfica y el visualizador son ejecutados localmente, mientras el servidor de base de datos y el motor de renderizado paralelo corre en un supercomputador.[16]

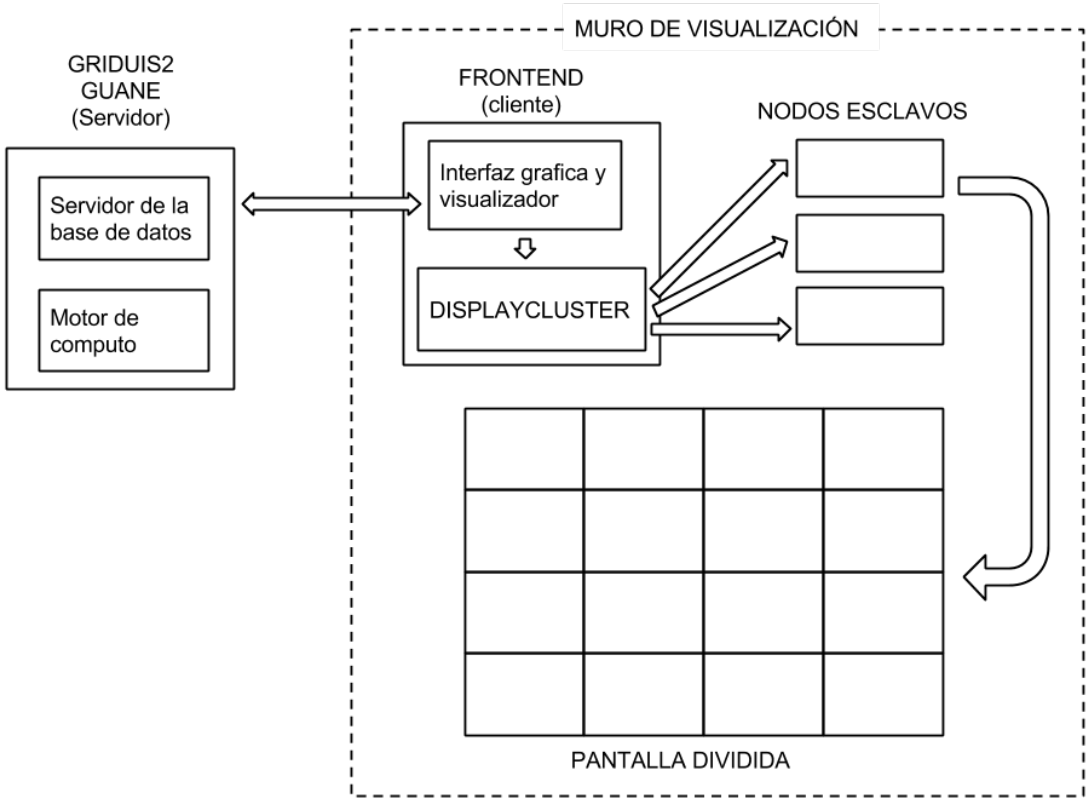
Por lo anterior, para el uso de la aplicación se planteo el utilizar una arquitectura cliente-servidor; dado el tamaño de los archivos, se obtienen las siguientes ventajas respecto a realizar el renderizado localmente:

1. No es necesario copiar los archivos a donde se desea visualizarlos, ya que los archivos son generados en la arquitectura *GRIDUIS2*, y en la misma arquitectura se realizaría el renderizado de la visualización

2. Los recursos para realizar el renderizado de la visualización son mayores, lo que permite visualizar archivos de mayor tamaño, que debido a limitaciones de memoria y/o computo gráfico no seria posible en un computador normal.

El modelo de la arquitectura cliente-servidor implementada, su funcionamiento, e interacción entre los diferentes dispositivos: el supercomputador guane (servidor), el frontend del muro de visualización (cliente), y los computadores esclavos del muro, se observa en la figura 14

Figura 14: Diagrama de la arquitectura cliente servidor, se puede observar la interacción de los diferentes componentes, y la interacción con el muro de visualización.



Para la configuración de la arquitectura se creo un ambiente de despliegue para ser usado en la plataforma *GRIDUIS2*, en el cual se realizo la compilación del software de visualización VisIt para lograr la ejecución en paralelo y poder aprovechar todos los recursos del nodo, también se configuro un sistema de archivos compartido NFS para usar el almacenamiento global de la plataforma y no tener que mover los archivos al nodo, y por ultimo se instalo el plugin desarrollado para la lectura de datos.

En el cliente se instalo desde los paquetes precompilados, debido a que no era necesario la compilación, porque no se requería que funcionara de forma paralela, ya que no

habría una mejora significativa porque el computo realizado en el computador cliente es menor.

Para las pruebas en un muro de visualización extendida, se uso el que actualmente dispone el grupo Halley, se realizo la instalación en el frontend, y mediante el software DisplayCluster [22] se genero la visualización sobre el muro.

[22]DisplayCluster es un ambiente de software para manejar interactivamente pantallas divididas de gran escala, https://www.tacc.utexas.edu/research-development/tacc-software/displaycluster [En linea][Consultado el 13 de febrero del 2015]

8 DESARROLLO DEL PLUGIN DE VISIT PARA LEER DATOS DIRECTAMENTE

Para lograr la visualización de datos en el software de visualización VisIt, la guía *Getting Data into VisIt* propone tres estrategias, crear archivos compatible, escribir un plugin para lectura de la base de datos, e instrumentar el código de la simulación. De las cuales se seleccionó la segunda de acuerdo a las razones antes mencionadas.

8.1 ESTRUCTURA DE VISIT

VisIt es una aplicación paralela y distribuida que consiste en cuatro componentes de procesado que trabajan conjuntamente para producir su visualización, el cliente, el visualizador, el servidor de base de datos y el motor de computo. [15] (en la figura 15 se puede observar la interacción entre estos componentes).

8.2 FUNCIONAMIENTO DE LOS PLUGIN DE LECTURA DE BASE DE DATOS DE VISIT

Un plugin de base de datos esta hecho por tres librerías compartidas, que son dinámicamente cargadas por los componentes apropiados de VisIt cuando los datos desde un archivo deben ser leídos. Los componentes de VisIt involucrados en leer datos desde un archivo son el servidor de bases de datos y el motor de computo. Cada plugin de lectura de base de datos tiene un componente servidor de base de datos, un componente de motor de computo, y un componente independiente para un total de tres librerías compartidas. [15]

8.3 INTERFAZ DEL PLUGIN DE LECTURA DE BASE DE DATOS

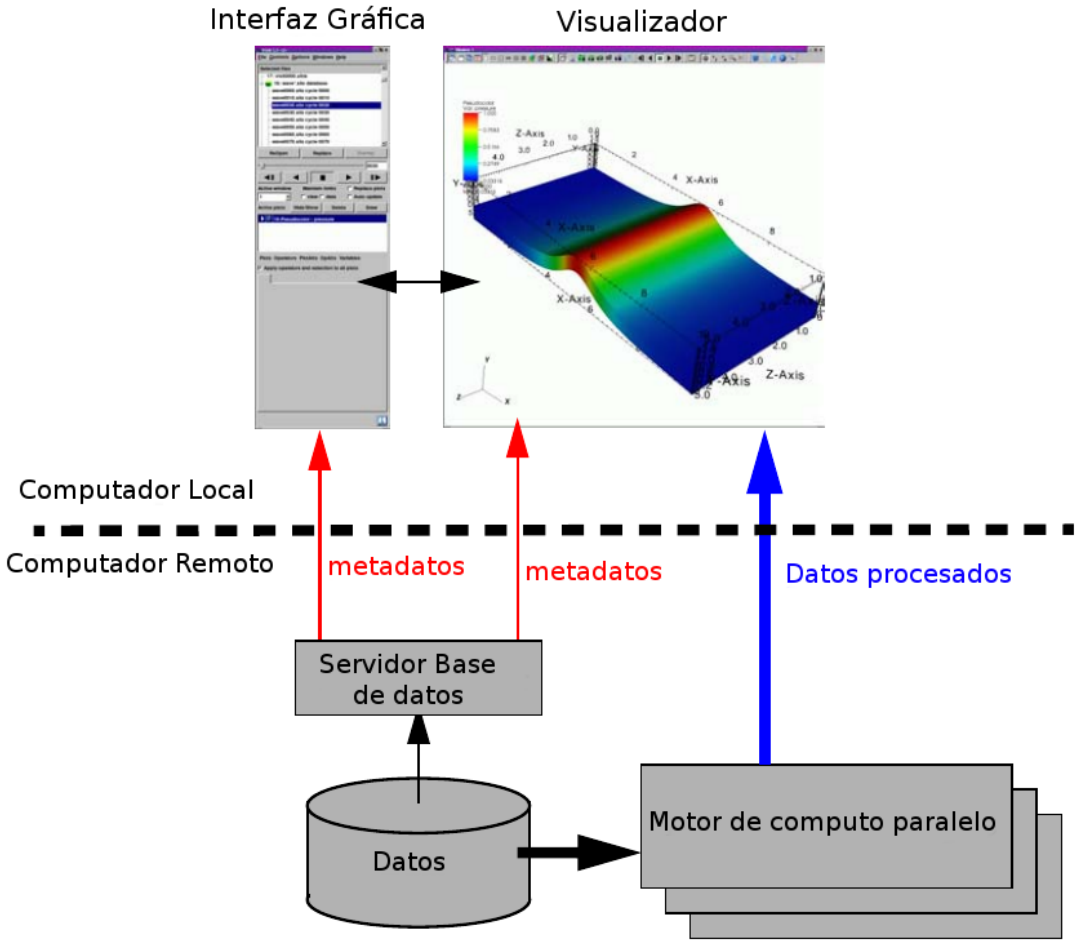
Los plugin de lectura de base de datos tienen 4 posibles interfaces, las cuales afectan como los archivos son mapeados a los objetos creados por el plugin (la tabla 2 muestra las cuatro posibles interfaces).

Para el desarrollo del plugin inicialmente se considero la interfaz STSD (Single time - Single Domain) para algunas implementaciones iniciales y luego se migro a STMD (Single Time - Multiple Domain), esta se considero la más adecuada ya que los archivos no representan múltiples estados temporales, sino un único estado incluyendo todas las trayectorias realizadas por las partículas, y se utilizaron múltiples dominios mediante la división de manera arbitraria, dada la total independencia de cada entrada del archivo entre sí, y de esta forma permitir que el computo posterior se realizara en paralelo.

8.4 FLUJO DEL FUNCIONAMIENTO DEL PLUGIN DE LECTURA DE BASE DE DATOS

Hay cuatro métodos principales que son necesarios para el funcionamiento del plugin. Estos métodos se encargan de la lectura de datos, la creación de estructuras de

Figura 15: Diagrama de la arquitectura del visualizador CORSIKA, se observa como el servidor de bases de datos y el motor de computo pueden ser ejecutados en un computador remoto.



Fuente: Adaptado del manual de lectura de datos de VisIt[15]

Tabla 2: Posibles interfaces de los plugin de base de datos, de acuerdo a los estados y dominios presentes en cada archivo de datos.

	Single Domain (SD)	Multiple Domain (MD)
Single time (ST)	STSD - Un solo estado temporal y contiene solamente un dominio.	STMD - Un solo estado temporal pero cada archivo contiene múltiples dominios.
Multiple time (MT)	MTSD - Múltiples estados temporales por archivo pero cada archivo contiene solo un dominio.	MTMD Múltiples estados temporales en cada archivo y cada archivo contiene múltiples dominios.

Fuente: Adaptado del manual de lectura de datos de VisIt[15].

la librería vtk, y retornarlos para que sean usados por el motor de computo y demás componentes.(tabla 3)

Tabla 3: Métodos principales usados en la interacción del plugin con el visualizador VisIt, se encargan de la lectura de datos y creación de estructuras VTK que son retornadas al visualizador para su posterior renderizado.

Método	Propósito
PopulateDatabase MetaData	Retorna una lista de mallas, variables escalares,variables vectoriales, y demas objetos contenidos en el archivo.
GetMesh	Retorna una malla, mediante una estructura de conjuntos de datos de VTK.
GetVar	Retorna los valores de los datos del archivo, en estructuras derivadas de tipo vtkDataArray.
GetVectorVar	Al igual que GetVar retorna valores de datos del archivo, pero en estructuras vtkFloatArray, con más de un valor por tupla.

Fuente: Adaptado del manual de lectura de datos de VisIt[15].

8.5 ESTRUCTURAS DE DATOS UTILIZADAS

Internamente el plugin, en la función `Initialize` lee los datos desde el archivo y crea una estructura de arrays conteniendo los datos y cambiando su organización; ya que el posterior manejo de los datos sera mejor que si se crea un array de estructuras. Debido al coste computacional esta función solo se realiza una vez y solamente cuando son requeridos los datos para evitar sobrecargar innecesariamente la aplicación.

Para la posterior visualización, se crean diferentes mallas, y estructuras VTK cuando son requeridas por el visualizador, estas estructuras son:

- Malla no estructurada `(vtkUnstructuredGrid)` representando las trayectorias
- Malla no estructurada `(vtkUnstructuredGrid)` representando los puntos iniciales y finales
- Arrays de flotantes `(vtkFloatArray)` o enteros `(vtkIntArray)` para las diferentes variables, energía, tiempo inicial y final, e identificación de la partícula.

Las mallas son no estructurados dadas las características de los datos, pues cada punto es guardado independientemente y sus coordenadas no tienen relación directa entre si, que permita representarlas mediante un grid estructurado o adaptativo.

9 PRUEBAS Y ANÁLISIS DE RESULTADOS

Para la realización de las pruebas se utilizaron los recursos de la arquitectura de computo *GRIDUIS2* del Centro de Supercomputación y Cálculo Científico (SC3UIS), y el muro de visualización extendida del Grupo Halley de Astronomía y Ciencias Aeroespaciales.

Para las pruebas realizadas en el supercomputador *GUANE* se utilizo un nodo de computo, con Sistema operativo Debian [23], procesador Intel Xeon E5640 de $2.67GHz$, memoria RAM $104GB$, y 8 tarjetas gráficas Nvidia Tesla M250, que utiliza un sistema de archivos NFS de $2TB$ compartido por la plataforma.

Para las pruebas locales se utilizo un computador de escritorio, con un Sistema operativo Debian, un procesador Intel Core I7-4790 de $3.6GHz$, memoria RAM de $8GBs$, y sin tarjeta gráfica dedicada.

Para las pruebas de visualización extendida se uso el muro de visualización del grupo Halley compuesto por un nodo maestro y cuatro esclavos, y una pantalla de gran escala conformada por 16 pantallas, en un grid de $4x4$.

9.1 COMPARACIÓN DE TIEMPO EN DOS MODELOS DE ARQUITECTURAS DIFERENTES

Para esta prueba se busco comparar el uso de una arquitectura de computo de alto desempeño usada remotamente, con un computador local de características normales.

En la figura 16 se puede observar el tiempo que se tardo en realizar una gráfica de una parte de una lluvia, para diferentes tamaños de archivo, utilizando una arquitectura remota y una local, se aprecia que debido a la latencia de la red el visualizar remotamente suele ser un poco mas demorado, pero la visualización remota permite manejar archivos de mayor tamaño, que la local no puede realizar por falta de recursos.

9.2 ESCALABILIDAD, CANTIDAD DE DATOS

Al usar plataformas de alto desempeño mediante la arquitectura cliente-servidor, se permitió el uso de archivos de mayor tamaño de los usados en un computador local. De los resultados de la figura 16 es posible observar que en la arquitectura local no fue posible realizar la visualización debido a limitantes de memoria.

Usando un nodo de computo del supercomputador Guane fue posible lograr visualizar archivos de tamaño del orden de los GBs generados por CORSIKA, lo cual era el objetivo del proyecto.

9.3 COMUNICACIÓN, CONSUMO DE RED

El uso de plataformas remotas tiene como principal desventaja el consumo de red, la figura 17, muestra dicho consumo para diferentes tamaños de archivos visualizados, utilizando la arquitectura cliente-servidor.

[23] https://www.debian.org [En linea][Consultado el 13 de febrero del 2015]

Figura 16: Comparación de tiempo necesario para obtener una visualización, se observa que debido a la latencia de red la arquitectura remota gasta un mayor tiempo. (Para el tamaño de archivo de $1GB$ no fue posible realizar la visualización usando el computador local, ya que el renderizado requería mayor memoria de la disponible).

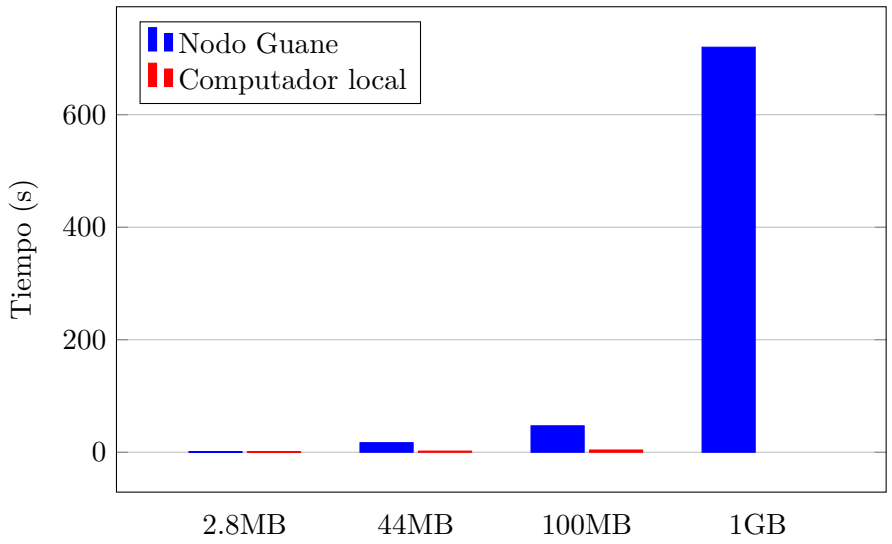
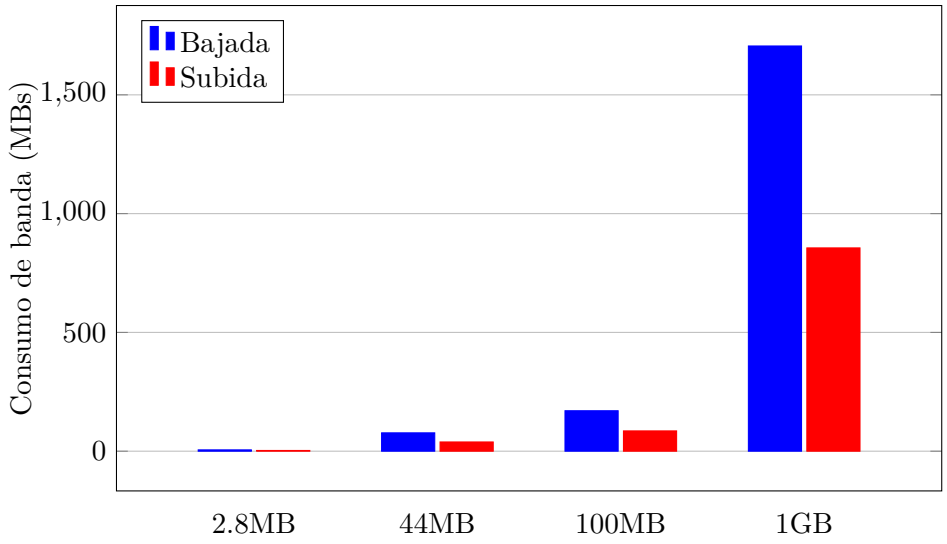


Figura 17: Consumo de red para diferentes tamaños de archivos visualizados, usando un computador remoto como servidor de renderizado, los valores de subida y bajada fueron calculados desde la perspectiva del cliente.



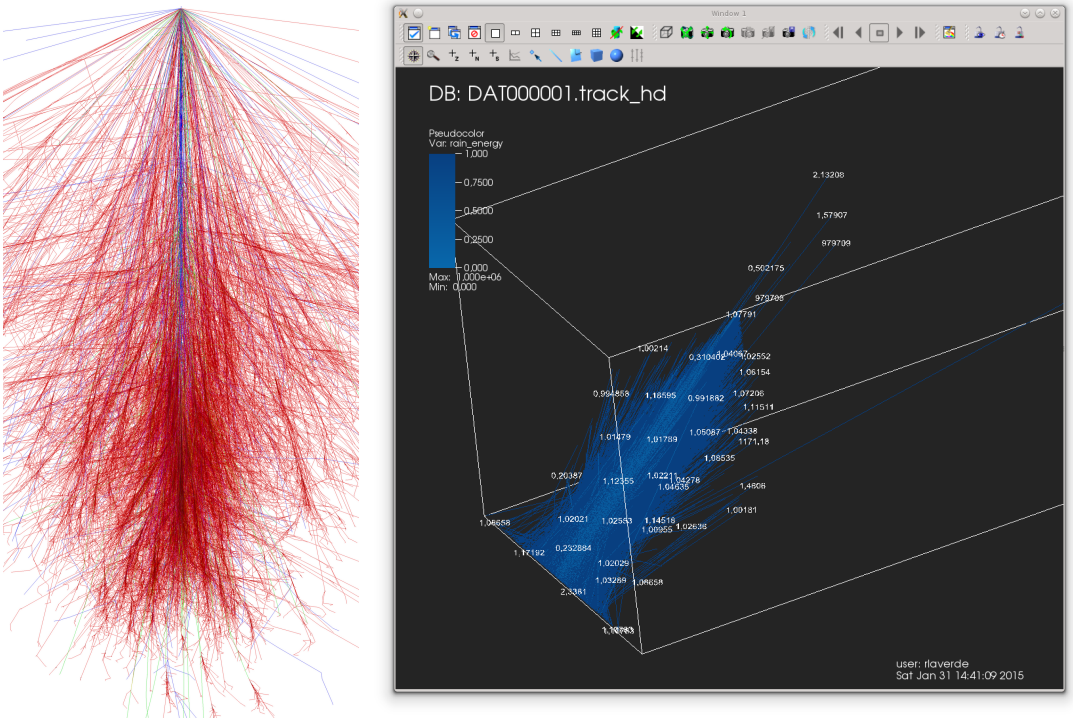
Estas pruebas se realizaron utilizando remotamente un nodo de computo del supercomputador Guane como servidor, y el otro computador descrito anteriormente como cliente. Para las mediciones del consumo de red se uso la diferencia entre el antes y después de los valores del estado de red disponibles en el archivo `/proc/net/netstat` el cual tiene diferentes estadísticas de red, para este caso se utilizaron *InOctets y OutOctets* las cuales indican la cantidad de bytes de subida y bajada respectivamente.

De estas gráficas podemos inferir que el consumo de red, se comporta directamente proporcional al tamaño de archivos, esto se presenta dado que aunque el renderizado, los filtros y transformaciones son calculados en el supercomputador, el estado final es trasmitido para poder interactuar con la visualización.

9.4 RESULTADOS DE LA VISUALIZACIÓN

Uno de los resultados mas significativos desde el punto de vista Astrofísico, es la mejora respecto a la visualización existente anteriormente, la figura 18 se realiza una comparación entre la visualización generada por las rutinas de CORSIKA y uno de las visualizaciones obtenidas con el desarrollo del presente proyecto.

Figura 18: Comparación de la visualización existente (izquierda), con la realizada en el presente trabajo(derecha), la visualización anterior representa la perspectiva xz de una lluvia; mientras la visualización lograda presenta el componente hadrónico de una lluvia utilizando falso color para representar los valores de energía.

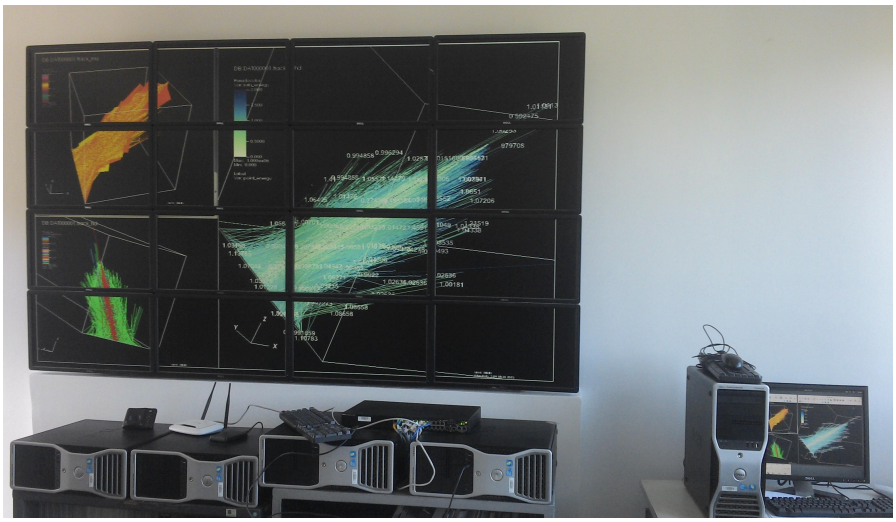


Fuente: (Imagen izquierda) Galeria pagina web de CORSIKA [17]

También cabe mencionar que además de la visualizaciones de las cascadas que era el objetivo del proyecto, el desarrollo realizado también permite la realización de diferentes tipos de gráficas, como histogramas o puntos dispersos, de los valores contenidos en los datos, lo cual es útil para realizar otros tipos de análisis.

Para la presentación y utilización de la aplicación se planteo la posibilidad de utilizar una plataforma de visualización extendida, de esta forma se puede apreciar mejor los detalles de la lluvia, la figura 19 muestra la visualización en funcionamiento utilizando el muro de visualización del grupo Halley, de acuerdo a la interacción de los componentes planteada en la figura 14.

Figura 19: Imagen del muro de visualización del grupo Halley mostrando diferentes visualizaciones de lluvias de cascadas de rayos cósmicos, Se puede observar el *frontend* del muro a la derecha y el muro y los nodos esclavos a la izquierda. Las visualizaciones presentadas representan gráficas de falso color de la energía y el id de partícula para los componentes hadrónico y muónico de una lluvia.



10 LIMITANTES DEL PROYECTO

Por la naturaleza del fenómeno de las cascadas de rayos cósmicos, los datos no pueden ser generados en un grid estructurado o adaptativo, lo cual implica el tener que guardar las coordenadas de cada punto de la malla, esto influye considerablemente en el tamaño de los archivos generados, respecto a un fenómeno que pueda ser simulado usando grid estructurado para representar los datos.

Por la forma de funcionamiento de simulación usada por CORSIKA, los datos no son generados en intervalos de tiempo definidos, sino que representan las trayectorias de las partículas y su tiempo inicial y final, por lo anterior no es posible obtener *snapshots* para realizar una animación del avance de la lluvia en el tiempo.

Se encontraron diversas dificultades al intentar representar los identificadores de las partículas, no como los números que usa CORSIKA internamente para su identificación, sino con los nombres correspondientes a cada partícula; debido a que el consumo de recursos es mayor, este tipo representación no es muy usado, y no se encontró suficiente documentación. Para lograr la diferenciación de las partículas se usaron gráficas de pseudo-color una técnica más usada en estos casos.

11 LINEAMIENTOS PLANTEADOS PARA LA VISUALIZACIÓN DE CONJUNTOS EXTENSOS DE DATOS

Existen muchas aplicaciones que requieren facilitar el análisis de los datos mediante visualización, proyectos que carecen del tiempo y recursos necesarios para implementaciones especificas y necesitan poder visualizar el fenómeno representado por las simulaciones o experimentos.

Para la obtención de visualizaciones de conjuntos extensos de datos, se plantean las siguientes etapas a seguir:

- Analizar la arquitectura de la aplicación que genera los datos a visualizar, con el objetivo de determinar las limitantes en un posterior acoplamiento.
- Analizar las diferentes herramientas de visualización que permiten el aprovechamiento de recursos de alto rendimiento, y determinar cual se acopla mejor a las necesidades de la visualización.
- Realizar los componentes necesarios para la estrategia escogida.
- Validar los resultados con las características del fenómeno, y los resultados esperados.

12 RECOMENDACIONES

Para trabajos posteriores en visualización de datos se recomienda utilizar herramientas de visualización disponibles como Paraview o VisIt y experimentar con las diferentes formas de lograr que la herramienta pueda utilizar los datos del simulador.

Para visualización de conjuntos extensos de datos se recomienda no realizar la conversión de los datos a otros formatos, dado que por limitantes de espacio no siempre es posible, aunque la conversión puede ser un buen inicio si no se conoce con claridad la estructura de los archivos, o se quiere obtener una idea de los posibles resultados a obtener.

Cuando se conozca mejor el funcionamiento de la aplicación y se pueda modificar el código fuente sin muchos inconvenientes, se aconseja usar la estrategia de que el simulador escriba los datos en un formato estándar, o la de instrumentar la aplicación de simulación.

Para la instalación y configuración de los componentes se recomienda realizar su compilación solo en el computador donde sera ejecutado el lector de base de datos y el motor de renderizado, ya que el proceso de compilación suele ser largo y dar algunos inconvenientes, y la mejora no sera significativa si es un cliente que se conecta a un supercomputador remoto.

En el caso particular de la visualización de cascadas de rayos cósmicos se puede experimentar realizando la subdivisión de los dominios usando los tipos de partícula, aunque el rendimiento podría ser menor, esta subdivisión tendría más sentido desde el punto de vista de la naturaleza de los datos.

13 CONCLUSIONES

En desarrollos anteriores de visualizaciones del fenómeno de cascadas de rayos cósmicos simulado por el software CORSIKA no se había planteado una visualización detallada de los datos usando arquitecturas de alto desempeño. Con la realización de este proyecto se logró obtener una visualización para los conjunto de datos extensos representando el fenómeno de las cascadas de rayos cósmicos, con un mayor nivel de detalle, que permite diferenciar cada partícula, y observar de una manera más clara y acorde el fenómeno representado por los datos de dicha simulación.

El software de simulación CORSIKA, no tiene una preocupación real por crear una arquitectura, debido a la forma en que fue desarrollado, su complejidad, múltiples dependencias de otros modelos de simulación, y el planteamiento limitado que en un inicio se le dio a su posible comunidad de usuarios.

Mediante el proceso de análisis de la arquitectura de funcionamiento de CORSIKA se logro identificar los problemas o limitantes, y plantear una estrategia de visualización acorde a estos, que permitiera obtener la visualización requerida.

Las aplicaciones de visualización de datos necesitan recursos computacionales de alto desempeño, para la visualización planteada, se propuso e implemento una arquitectura cliente-servidor, realizando el renderizado en un supercomputador y visualizando remotamente, además de utilizar un muro de visualización para apreciar mejor los detalles de la visualización.

Con la realización de este proyecto se logró proponer lineamientos para el planteamiento de estrategias, y construcción de aplicaciones de visualización de conjuntos de datos extensos, aprovechando recursos computacionales de alto desempeño.

A partir de la visualización extensiva de datos se puede obtener un mayor nivel de detalle, y en el caso particular de las cascadas de rayos cósmicos, distinguir las diversas interacciones de las partículas de la lluvia, permitiendo un análisis más especifico de regiones de la lluvia; con los métodos actualmente usados para la visualización de dicho fenómeno solo es posible realizar análisis generales del comportamiento de la lluvia.

La interacción con plataformas de visualización extendida, se puede realizar mediante *Pixel Streaming*, utilizando software disponible actualmente, como *DisplayCluster*.

Para la visualización de conjuntos extensos de datos, y algunos problemas que requieren manejos de grandes cantidades de datos, es más adecuado la utilización de estructura de arrays, ya que permite un acceso más eficiente a la memoria cuando se requiere leer los datos correspondientes a una variable.

Las herramientas de visualización científica de datos, presentan un grado de desarrollo considerable, diversas características para el análisis y visualización de diferentes estructuras de datos, pueden ser ejecutadas en paralelo aprovechando los recursos de plataformas de computo de alto desempeño, y permiten ser acopladas y utilizadas para diversos casos específicos de visualización, lo cual fue clave para el presente proyecto.

El desarrollo de un plugin para VisIt requiere la identificación del tipo de interfaz que se acopla al problema, y el uso de las estructuras adecuadas de acuerdo a la naturaleza de los datos.

BIBLIOGRAFÍA

[1] James Ahrens, Kristi Brislawn, Ken Martin, Berk Geveci, Charles Law, and Michael Papka. Large-scale data visualization using parallel data streaming. *IEEE Computer Graphics and Applications*, pages 1–3, 2001.

[2] Hernán Asorey. Los detectores cherenkov del observatorio pierre auger y su aplicación para el estudio de fondos de radiación, tesis doctoral. *Instituto Balseiro, UnCuyo*, 2012.

[3] Hernán Asorey. Cálculo de fluencia en lago, flujos primarios y fondo de secundarios. *Centro Atómico Bariloche (CNEA)*, 2:1–2, 2013.

[4] Hernán Asorey. Cálculo de fluencia en lago, instalacion de corsika. *Centro Atómico Bariloche (CNEA)*, 1:1, 2013.

[5] Hank Childs, Eric Brugger, Brad Whitlock, Jeremy Meredith, Sean Ahern, David Pugmire, Kathleen Biagas, Mark Miller, Cyrus Harrison, Gunther H. Weber, Hari Krishnan, Thomas Fogal, Allen Sanderson, Christoph Garth, E. Wes Bethel, David Camp, Oliver Rübel, Marc Durant, Jean M. Favre, and Paul Navrátil. VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, pages 357–372. Oct 2012.

[6] Heck D., R Engel, G Battistoni, A. Fasso, A. Ferrari, J Ranft, and P.R. Sala. Influence of low-energy hadronic interaction programs on air shower simulations with corsika. *n Proc. 28th ICRC*, 1:279, 8 2003.

[7] George S Gottlieb Allan; Almasi. Highly parallel computing. *Redwood City, California.: Benjamin/Cummings*, 1989.

[8] Amr Hassan and Christopher J. Fluke. Scientific visualization in astronomy: Towards the petascale astronomy era. *Publications of the Astronomical Society of Australia (PASA)*, pages 1–6, 2011.

[9] D. Heck, J. Knapp, J.N. Capdevielle, G. Schatz, and T. Thouw. *CORSIKA: A Monte Carlo Code to Simulate Extensive Air Showers*. FZKA, 1998.

[10] D. Heck, J. Knapp, J.N. Capdevielle, G. Schatz, and T. Thouw. *Extensive Air Showers Simulation with CORSIKA*. FZKA, 1998.

[11] Dieter Heck. Corsika an air shower simulation program [en linea]. `http://www-ik.fzk.de/~corsika/`, [Consultado el 13 de febrero del 2015].

[12] Gregory P. Johnson, Gregory D. Abram, and Brandt Westing. Displaycluster [en linea]. `www.tacc.utexas.edu/tacc-software/displaycluster/`, [Consultado el 13 de febrero del 2015].

[13] Kitware. Paraview [en linea]. http://www.paraview.org/, [Consultado el 13 de febrero del 2015].

[14] Kitware. Visualization toolkit (vtk) [en linea]. http://www.vtk.org/, [Consultado el 13 de febrero del 2015].

[15] Lawrence Livermore National Laboratory. *Getting Data Into VisIt.* 07 2005.

[16] Lawrence Livermore National Laboratory. *VisIt User's Manual.* 10 2005.

[17] Lawrence Livermore National Laboratory. Visit gallery [en linea]. https://wci.llnl.gov/simulation/computer-codes/visit/gallery, [Consultado el 13 de febrero del 2015].

[18] Fabian Schmidt. Corsika shower images [en linea]. http://www-zeuthen.desy.de/~jknapp/fs/, [Consultado el 13 de febrero del 2015].

[19] SGI. Opengl overview [en linea]. www.opengl.org/about/, [Consultado el 13 de febrero del 2015].

[20] Sharma. *Atomic And Nuclear Physics.* Pearson Education India, 2008.

[21] Simon Swordy. Cosmic rays [en linea]. http://apod.nasa.gov/apod/ap060814.html, [Consultado el 13 de febrero del 2015].

[22] Visit user community. Visit user community web site, [en linea]. http://visitusers.org/, [Consultado el 13 de febrero del 2015].

ANEXO A: DIAGRAMA DE FLUJO ENTRE LOS COMPONENTES DE LA APLICACIÓN Y EL PLUGIN DE LECTURA DE BASE DE DATOS