Construcción de un túnel de viento virtual sobre AccessGrid y ParaView.

Camilo Cortés
Estudiante Maestría Ingeniería de Sistemas y Computación,
Universidad de los Andes
Carrera 1 #18ª-10
Bogotá, Colombia
05713334949 ext. 3462
cam-cort@uniandes.edu.co

Andrés Padilla
Estudiante Maestría Ingeniería de Sistemas y Computación,
Universidad de los Andes
Carrera 1 #18ª-10
Bogotá, Colombia
05713334949 ext. 3462
a-padill@uniandes.edu.co

Pablo Figueroa, Ph.D
Profesor Asociado Universidad de los Andes
Carrera 1 #18ª-10
Bogotá Colombia
05713334949 ext. 2864
pfiguero@uniandes.edu.co

Manuel García, Ph. D
Grupo de Mecánica Aplicada,
Universidad EAFIT
Carrera 49 No 7 sur 50
Medellín, Colombia
mgarcia@eafit.edu.co

Pierre Boulanger, Ph. D
AMMI Laboratory, Universidad de Alberta
ATH411, TG6 2R3
Edmonton, Alberta, Canadá
PierreB@cs.ualberta.ca

RESUMEN

Este documento introduce un nuevo sistema de visualización distribuido basado en herramientas libres que hace una implementación de los conceptos de *Virtual Steering*. Se mostrará la arquitectura del sistema así como los resultados obtenidos en el proceso.

Categories and Subject Descriptors

D.3.3 [**Programming Languages**]: Language Contructs and Features – *abstract data types, polymorphism, control structures.* This is just an example, please use the correct category and subject descriptors for your submission. The ACM Computing Classification Scheme: http://www.acm.org/class/1998/

General Terms

Algorithms, Measurement, Performance, Design, Experimentation.

Palabras Claves

Computación Gráfica y Realidad Virtual.

1. INTRODUCCIÓN

Los avances en sistemas de computación y redes de computadores de gran velocidad han permitido el surgimiento de nuevas técnicas de visualización sobre grandes conjuntos de datos que hace algunos años eran imposibles. Una de las aplicaciones que se ven potenciadas con el nuevo hardware disponible es la manipulación virtual (Virtual Steering) [1], que consiste en poder interactuar con una simulación científica en tiempo real, incluso cambiando los parámetros de simulación y los puntos de vista de la visualización de la simulación; junto con la posibilidad de que personas en localizaciones remotas puedan también cambiar parámetros de visualización por medio de una red de alta velocidad.

Asimismo, avances en computación de alto rendimiento (HPC, High Performance Computing) han permitido que simulaciones que antes sólo podían ser ejecutadas en modo batch debido a su alto costo computacional ahora puedan ver sus parámetros modificados en tiempo real. Una de las áreas donde se pueden encontrar este tipo de avances en HPC es el área de simulación de dinámica de fluidos (CFD, Computational Fluid Dynamics), donde es ahora posible realizar costosas simulaciones de objetos inmersos en fluido en tiempo real. Una aplicación de este tipo de simulaciones es un túnel de viento, aplicación muy atractiva por los recursos que pueden ser ahorrados al no tener que construir un túnel de viento físico.

En este artículo se muestra como se logra hacer la integración entre software y hardware con el propósito de obtener un sistema de visualización de alto rendimiento en un ambiente distribuido. Entre las herramientas que usamos, todas de código abierto están VTK [2] (Visualization Toolkit) para la visualización de los datos, ParaView [3] para el procesamiento en paralelo de archivos de simulación, AccessGrid [4] para transferencia de datos, comunicación y como framework para la construcción de aplicaciones distribuidas.

2. TRABAJOS EN EL ÁREA

Anteriormente se han desarrollado varios trabajos de interacción con visualizaciones para aplicaciones científicas. Entre ellos se encuentra una aplicación de Realidad Virtual desarrollada en la Universidad de Illinois para interactuar y manipular objetos provenientes de una simulación científica computada remotamente [5], un framework para desarrollo de túneles de viento virtuales desarrollado por la NASA [6] y su implementación [7].

El propósito del Túnel de Viento Virtual desarrollado por la Nasa es tener un sistema inmersivo en dónde el usuario pueda interactuar con distintos modelos y observar cambios en el flujo que lo circula alrededor. El empleo del *boom-mounted display*, una variante del conocido *head-mounted display* sobre el usuario y el uso de guantes son los elementos a resaltar en este sistema de visualización. El *boom-mounted display* permite al usuario un mayor grado de libertad y un menor grado de molestia ya que evita que la persona tenga que cargar un casco sobre su cabeza los cuales pueden llegar a ser pesados. El sistema entero está dado como un framework (libre) para que sea adoptado como sistema de visualización.

Por otro lado, la universidad de Illinois en Chicago desarrolló un ambiente inmersivo en un *CAVE* para la visualización de complejos cálculos computacionales sobre el fenómeno físico de los Agujeros de Gusano. Las ecuaciones detrás de este particular suceso estelar necesitan un gran poder computacional por lo cual la Universidad de Illinois dispuso de varios supercomputadores que se resuelven las intrincadas ecuaciones dando como resultados *timesteps* (tiempos de ejecución) que serán visualizados en el *CAVE*. Este último permite una visualización amplia y tridimensional del Agujero de Gusano usando la técnica de proyección en estéreo.

Como vemos diferentes técnicas de visualización e interacción son empleadas así como el uso de robustos *frameworks* de modelos computacionales y computación de alto rendimiento (HPC). Entre estos elementos se usan modelos de paso de mensajes (MP) para aprovechar el paradigma del paralelismo y maximizar el uso de los recursos, sistemas de visualización inmersivos como *CAVE,* redes de alta velocidad para transferencia de grandes volúmenes de datos, uso de hardware especializado para procesamiento paralelo (GPUs o Estaciones de trabajo especializadas). Nuestro trabajo, sin embargo, como veremos a continuación se basa en una arquitectura más colaborativa, en dónde no solo se tiene un lugar de visualización con requerimientos específicos sino encontramos una problemática de visualización distribuida en ambientes heterogéneos.

3. ARQUITECTURA

Antes de describir la arquitectura del sistema en sí y de cómo fue desarrollado es importante conocer las problemáticas que se plantearon para su concepción. El principal problema que se presentó era que se necesitaba construir un sistema de visualización distribuido de alto rendimiento. Para esto se necesitó esclarecer varios factores tanto de hardware como de software como lo son el tipo de red por la que circularían los datos, los equipos que se usarían para el procesamiento y visualización de los datos, que programas de código abierto para la construcción de aplicaciones se usarían teniendo en cuenta que fuesen compatibles con el hardware para obtener el mayor rendimiento posible. También fue crucial entender cómo se comunicarían las diferentes partes del sistema para diagramar el flujo de datos y poder construir el cliente que manejaría la visualización y comunicación de los conjuntos de datos. Con esto se quiere lograr la creación de un sistema de visualización colaborativo de gran capacidad de cómputo con el fin de simular las propiedades de un Túnel de Viento.

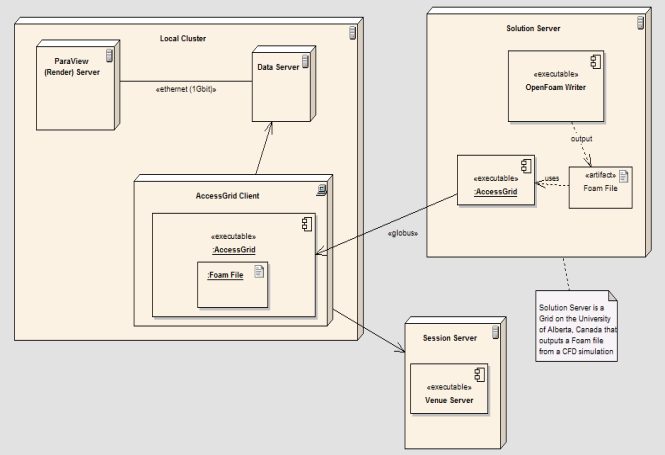

Figura 1: Arquitectura del sistema

La arquitectura comprende un sistema encargado de la ejecución y procesamiento de las simulaciones CFD (*Computer Fluid Dynamics)*, un sistema dedicado a la comunicación y paso del contenido a través de una red, otro sistema especializado en la visualización del contenido y un último sistema encargado de mantener la información de sesiones de los usuarios. La interacción de todos estos sistemas está descrita en el Diagrama de Despliegue (Figura 1).

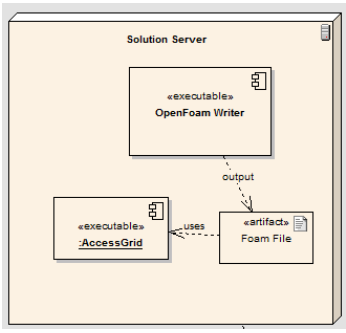

Figura 2: Sistema CFD

El sistema de procesamiento de CFD consiste en un grid para la resolución de un sistema de ecuaciones que modelan un objeto y el flujo de un fluido alrededor de él. Este se encuentra descrito como el *Solution Server* en la Figura 2. Como resultado de esta ejecución se obtiene un archivo de simulación en formato VTI (*Visualization Toolkit Image Data)*. Este archivo es el que se usará para la visualización en distintos ambientes.

Para la distribución del archivo de simulación se usa la herramienta libre AccessGrid que sirve como middleware entre un el sistema de procesamiento CFD y el sistema de visualización. Cada nodo de visualización tendrá una o varias máquinas clientes de AccessGrid dependiendo de su configuración local. Estas máquinas cliente se conectaran a una sesión o como se llama formalmente *Venue Server* que consiste en un servidor que mantiene el estado de una sesión AccessGrid. Allí se pueden alojar datos comunes a todos los participantes como sería el caso de los archivos de simulación VTI, para su posterior visualización. Además AccessGrid ofrece otros servicios como videoconferencias que son pertinentes para la realización de una aplicación colaborativa.

Para la visualización de contenido se utiliza como servidor de *rendering* la herramienta de visualización paralela desarrollado por KitWare llamada ParaView. Esta aplicación está compuesta por un servidor paralelo que usa MPI (*Message Passing Interface*) para la comunicación entre los distintos servidores paralelos (i.e. Los computadores de un cluster) y que usa VTK para la visualización de los datos. También incluye un cliente QT con distintas funciones de visualización y análisis de imágenes.

Existen dos modos de organización del servidor ParaView que se explican a continuación.

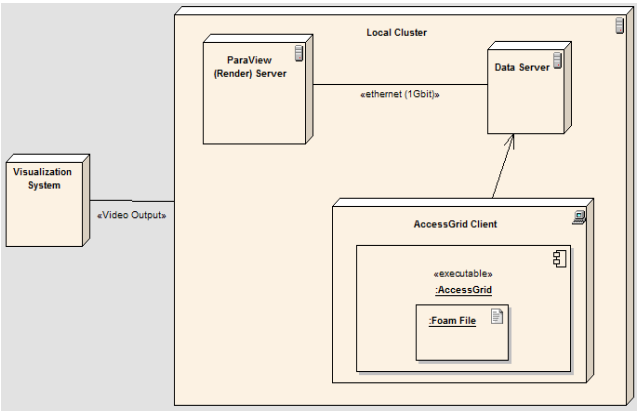

Figura 3: Clúster de visualización

Servidor Único (compuesto)

El servidor de *render* unido al servidor de manejo de datos en un único trabajo paralelo (*pvserver*) Es el modo estándar de funcionamiento del servidor de ParaView.

Servidor de Datos y Servidor de Render (desacoplado)

Los dos componentes anteriores corriendo en trabajos paralelos separados. Se utiliza para usar un equipo con una potente GPU para hacer el *render* y otro equipo con mejor capacidad de procesamiento para manejar la entrada y salida de los altos volúmenes de datos gráficos. Este procesamiento se hace a expensas de una mayor carga en la red debido a un alto flujo de datos entre el servidor de *render* y el servidor de datos (Data Server + Render Server). Kitware recomienda no usarlo debido al sobrecosto de partición de los datos.

El cluster usado para este trabajo está conformado de la siguiente manera: siete máquinas conectadas de la manera mostrada en la (figura 4):

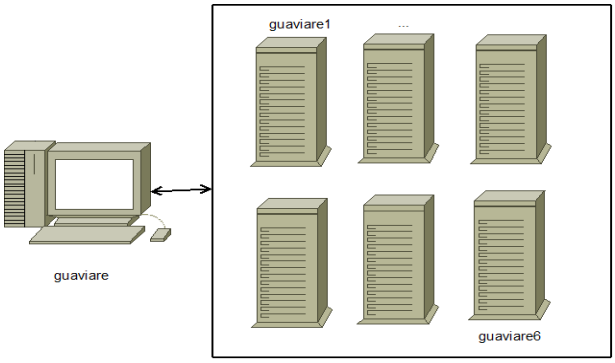

Figura 4: Clúster de Visualización usado

El clúster está conformado por siete equipos con características parecidas. Cada uno tiene un procesador de cuatro núcleos Intel Core 2 Quad a 3.00 GHz, 8 GB de Memoria RAM, dos discos duros de 500 GB y uno de 80 GB, y dos tarjetas de red Gigabit Ethernet. El sistema operativo escogido es CentOS versión 5, una versión abierta de Red Hat Enterprise Linux. El equipo Guaviare es la cabeza visible del cluster, ya que es el único que tiene conexión a Internet. El resto está aislado de la red externa, pero los equipos tienen una doble red interna de un gigabit. Guaviare se conecta a una sola de las redes internas, y de esta manera es posible enviar información hacia el exterior. Un problema de esta configuración es el cuello de botella que representa el servidor Guaviare.

3.1 Flujo de Datos

El flujo de datos del sistema se basa en tres pasos: Generar el archivo de simulación; enviar el archivo de simulación al clúster; visualizar el archivo en el ambiente escogido.

Todo el trabajo comienza en el sistema de procesamiento de ecuaciones CFD. Al ejecutar el conjunto de ecuaciones de Navier-Stokes para fluidos se obtiene un archivo que contiene los *timesteps* de la simulación. Luego, como se muestra en la Figura 1, el servidor AccessGrid recoge la información de los *timesteps* y lo envía a los clientes remotos. Después el cliente remoto AccessGrid le entrega la información al servidor de datos ParaView, que a su vez realiza la división de los datos y del trabajo entre los procesadores del clúster y así entrega el trabajo al servidor de *render* que se encargara de visualizar la información localmente con las opciones pertinentes a la forma de visualización utilizada. Las formas de visualización pueden ser un *CAVE*, una pantalla dividida (*tiled-display*) o una pantalla de proyección estéreo.

4. IMPLEMENTACIÓN ACTUAL

Lo que se tiene en el momento es un prototipo de la aplicación colaborativa y varias pruebas sobre el rendimiento y usabilidad de ParaView.

4.1 Aplicación Colaborativa

La aplicación colaborativa AccessGrid esta implementada en python [8] y su interfaz construida con ayuda de wxPython [9], que es un API para la construcción de interfaces en python. La aplicación, llamada AGVWT (AccessGrid Virtual Wind Tunnel) ofrecerá las siguientes características:

- Conexión transparente al servidor de *rendering*
- Participación en modo Maestro/Esclavo o Libre
- Cargar simulaciones OpenFOAM [10]
- Visualización de la simulación

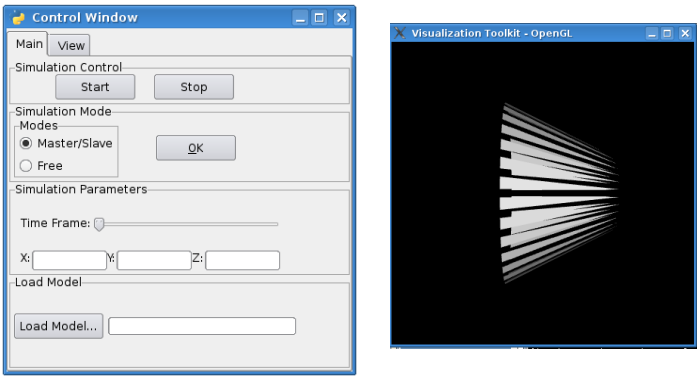

Figura 5: Ventana de Control con objeto

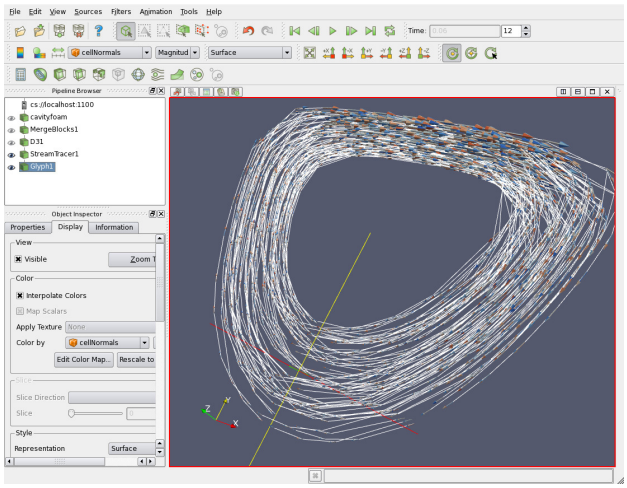

Figura 6: Datos de un flujo visualizados en Paraview

De la aplicación se estudiaron dos vertientes de implementación: Una iba a ser desarrollada en wxPython y accedería los servicios de ParaView a través del ServerManager, que es un *wrapping* en python de las librerías de ParaView. La otra es una aplicación construida en VTK (Visualization Toolkit) que usaría las librerías de ParaView directamente para hacer el *rendering* paralelo de las simulaciones.

4.1.1 Aplicación ServerManager con wxPython

En la primera aplicación se logró un diseño a manera de borrador de lo que sería la interfaz y se hizo la conexión a ParaView con el ServerManager. Lastimosamente esta extensión no permite hacer una sesión interactiva con el servidor de *render* de ParaView por lo cual lo único que se logra es la visualización del modelo. En otras palabras no se puede interactuar con el objeto en pantalla. La Figura 5 muestra la ventana de control y un modelo cónico. Principalmente la aplicación se construyó sobre AcessGrid para lograr la distribución del contenido fácilmente sobre los diferentes participantes (Canadá y dos ubicaciones en Colombia). En un *Venue* local que es el lugar a dónde todos los participantes se conectan se instaló la aplicación para que está pudiera ser accedida y usada como una aplicación por defecto de AccessGrid. Así, el aplicativo permite cargar un modelo en formato VTK (será integrado posteriormente con archivos FOAM) que se encuentra guardado en el *Venue* y visualizarlo. Dependiendo del modo de conexión, si es Maestro todos los cambios al modelo actual, como cambio de punto de vista son propagados a todos los esclavos (aquellos participantes que se conectan cuando ya hay una sesión abierta). Más importante aún la aplicación permitirá comenzar una simulación y pararla (Start/Stop) convenientemente.

4.1.2 Aplicación VTK

Paralelamente se desarrolló una aplicación estrictamente VTK que tendría como objetivo comunicarse con ParaView directamente. En este momento la opción está en discusión ya que significaría reinventar la rueda puesto que se tendría que implementar los servicios de conexión y de procesamiento paralelo que ParaView ya tiene.

4.2 Visualización Paralela con ParaView

De la misma manera se han hecho varios trabajos con ParaView. Lo primero fue revisar la configuración de la aplicación para poder correrla en paralelo en el clúster disponible. Se logró correr el servidor paralelo (pvserver) en las seis (6) máquinas del clúster. Aunque primero se hicieron pruebas con ParaView 3.2.3, con la experiencia obtenida logramos correr ParaView 3.4 sin tantos problemas. Después se intentó usar instalar el lector estándar de OpenFOAM (librería usada para calcular y mostrar los resultados de la simulación CFD), pero no funcionó después de ser compilado. Recurrimos entonces a un lector alternativo, que además tiene la ventaja de ser paralelo, desarrollado por el Sr. Takuya Oshima en la Universidad de Niigata [11]. Con este lector pudimos leer los archivos de OpenFOAM y visualizarlos con éxito. Realizamos algunas pruebas de rendimiento con un caso de prueba de 2 GB. Los resultados son resumidos en la tabla 1. Los parámetros de configuración del objeto ParaView de la prueba pueden ser vistos en la Figura 7.

Número Procesadores	Tiempo de ejecución
4	25.51 s
8	33.25 s
12	42.03 s

Tabla 1: Tiempo empleado por el clúster para mostrar 30 *timesteps* de la simulación con el lector paralelo.

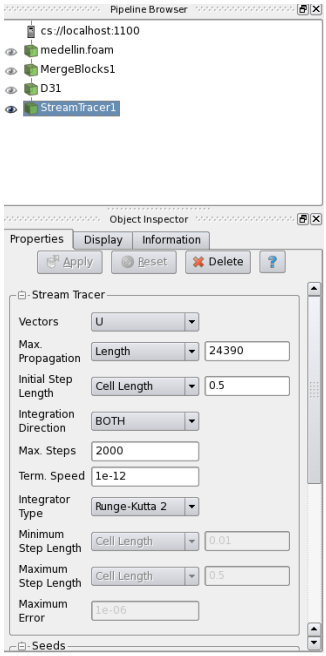

Figura 7: Configuración cliente ParaView para la prueba de ejecución.

Como se puede observar en la tabla, todavía no se ha optimizado el funcionamiento del lector paralelo lo suficiente como para sacar provecho de los recursos computacionales disponibles, esta es un área donde se pueden hacer mejoras de rendimiento.

5. TRABAJO FUTURO

El siguiente paso en el desarrollo es crear un plugin que se adapte a la interfaz gráfica de ParaView, para dar control sobre la visualización del objeto de interés y para comunicar los cambios de visualización al cliente de AccessGrid. También se quiere integrar el ServerManager con el trabajo hecho en VTK para lograr interactuar con el objeto en pantalla. Por último, se espera configurar mejor el clúster para lograr el rendimiento que se espera del hardware.

6. REFERENCES

[1] Garcia, M et al. Túnel de Viento Virtual. Obtenido de http://renata.edu.co/index.php/descargas/doc_download/33-tunel-de-viento-virtual.html el 14 de enero de 2009.

[2] Visualization Toolkit, VTK. Obtenido de http://www.vtk.org el 14 de Enero de 2009.

[3] Kitware, Paraview. Obtenido de http://www.paraview.org el 14 de Enero de 2009.

[4] AccessGrid 3.1. Obtenido de http://www.accessgrid.org el 14 de Enero de 2009.

[5] Roy, T. and Cruz-Neira, C. and DeFanti, T. 1995. Cosmic Worm in the Cave: Steering a High Performance Computing application From a Virtual Environment. Presence: Tele-operators and Virtual Environments. University of Illinois at Chicago.

[6] Bryson, S. and Levit, C. 1991. The Virtual Windtunnel: An Environment for the Exploration of Three-Dimensional Unsteady Flows. Proceedings of the 2nd conference on Visualization '91.

[7] The Virtual Windtunnel. Obtenido de http://www.nas.nasa.gov/Software/VWT el 14 de Enero de 2009.

[8] Python Programming Language. Obtenido de http://www.python.org el 14 de Enero de 2009

[9] WxPython Library. Obtenido de http://www.wxptyhon.org el 14 de Enero de 2009.

[10] The Open Source CFD Toolbox, OPENFOAM. Obtenido de http://www.opencfd.co.uk/openfoam el 14 de Enero de 2009.

[11] Contrib Parallelized Native OpenFOAM Reader for ParaView. Obtenido de http://openfoamwiki.net/index.php/Tip_Building_ParaView3_With_OpenFOAM_Native_Reader el 14 de Enero de 2009.