Visualización de fluidos mediante técnicas en dinámica de fluidos computacionales sobre máquinas masivamente paralelas basadas en GPUs

César Alexander Cesar Bernal Díaz

Trabajo de Grado para Optar el título de Ingeniero de Sistemas e Informática

Director
Carlos Jaime Barrios Hernández
Doctor en Informática

Codirector
Jorge Luis Chacón Velasco
Doctor en Ingeniería Mecánica

Universidad Industrial de Santander
Facultad de Ingenierías Físico-Mecánicas
Escuela de Ingeniería de Sistemas e Informática
Bucaramanga
2018

Dedicatoria

A Dios, quien siempre está presente y mantiene viva la llama del deseo de superación y siempre pone las personas adecuadas en mi camino; a él primeramente dedico mi trabajo.

De igual forma dedico mi trabajo de grado a mi madre quien es el motor de mi vida, quien siempre está presente para brindarme un consejo y palabras de alivio en los momentos más difíciles.

A mi hermanita quien ha estado junto a mí, apoyándome y me demuestra cada día y aunque sea mejor de yo, puede llegar a ser una persona muy madura y digna de admirar.

Agradecimientos

La elaboración de este trabajo fue posible gracias a la valiosa colaboración del Doctor Carlos Barrios, quien gracias a su seguimiento académico logra mantener enfocados a los estudiantes en sus objetivos profesionales.

A Cecilita, anterior secretaria de la escuela quien me dedico las palabras “Dios nunca te abandonará, sigue adelante” y me apoyo enormemente, me guio en todos los procesos para regresar a la vida universitaria.

A mi preciado amigo Camilo quien siempre me aconsejo no renunciar e hiciere el último esfuerzo por ser alguien en la vida.

Al profesor Gilberto Díaz quien con paciencia y experiencia ha aportado a mi desarrollo personal y profesional y es para mí, un punto de referencia para seguir siempre aprendiendo y aportándole a la comunidad.

Al profesor Jorge Chacón quien siempre me extendió una mano amiga y me guio para poder desarrollar un proyecto de carácter interdisciplinar.

En general a todos mis profesores de los cuales he to tomado las mejores enseñanzas, que me han moldeado para ser un mejor profesional.

A mi familia por su paciencia y apoyo para poder culminar mi desarrollo profesional.

Tabla de Contenido

Pág.

Lista de Tablas

Lista de Figuras

Lista de Apéndices

(Ver apéndices adjuntos en el CD y pueden visualizarlos en la Base de Datos de la Biblioteca UIS)

Glosario

Cgroups: Abreviatura de Grupos de Control, una característica del kernel Linux que limita recursos como CPU, memoria, disco, I/O para cada proceso. *Se define como, cuanto de esos recursos computacionales puede usar cada proceso.*

GUANE: clúster de computadoras de alto rendimiento, GpUs AdvaNced Environment por sus siglas en ingles.

Microservicios: Es una metodología de implementación para arquitecturas orientadas como servicio usada para desplegar sistemas de software flexible e independiente.

Namespace: también llamada virtualización de procesos ligeros, característica del kernel Linux, que permite aislar grupos de procesos para tener diferentes vistas del Sistema Operativo; *se define como, que puede ver cada proceso*.

NFS: (Network File System) Sistema de archivos en red, metodología de acceso a recursos de almacenamiento de información de forma centralizada a través de una red de interconexión entre computadoras.

Passthrough: término utilizado para indicar que un software cliente virtual tiene acceso total al hardware PCI para un rango amplio de tareas como si el dispositivo estuviese conectado directamente al sistema operativo.

Proxy: Software o dispositivo de hardware que se sitúa en medio de dos procesos, o equipos de cómputo, tiene por objeto servir como elemento intermedio de comunicación proveyendo alguna funcionalidad adicional.

Renderizado: Termino que refiere al cálculo necesario realizado por un procesador para representar información de forma gráfica en una pantalla.

RFB: Remote Frame Buffer, protocolo de intercambio de marcos de información gráfica (pixeles)

Solver: algoritmos de software que tienen como función resolver algebraicamente funciones discretizadas de diversos campos ingenieriles.

TLS: Long Term Support, nomenclatura que se le asigna a las versiones del sistema operativo Ubuntu a las cuales se les da soporte técnico por un periodo prolongado de tiempo.

Tunel SSH: metodología de acceso a equipos de cómputo localizados tras un firewall corporativo, de forma cifrada, también llamada redireccionamiento de puertos.

Socket Unix. Puerto de comunicación entre procesos, tiene la finalidad de mecanismo de intercambio de datos entre procesos que se ejecutan en un mismo sistema operativo.

VNC: Virtual Network Computing, tecnología de redes virtuales que permite visualizar escritorios remotos.

Web sockets: Protocolo de comunicación entre computadoras bidireccional, opera sobre TCP, *RFC 6455 del 2011*.

X server: Software que se encarga de procesar las llamadas de una aplicación a la API openGL, crea un contexto de trabajo para que la aplicación pueda hablar directamente con el hardware de video.

RESUMEN

TITULO: VISUALIZACIÓN DE FLUIDOS MEDIANTE TÉCNICAS EN DINÁMICA DE FLUIDOS COMPUTACIONALES SOBRE MÁQUINAS MASIVAMENTE PARALELAS BASADAS EN GPUS [1]

AUTOR: CESAR ALEXANDER BERNAL DIAZ[2]

PALABRAS CLAVE: CLÚSTER, HPC, GPU, CONTENEDORES LINUX, VISUALIZACIÓN CIENTÍFICA, VIRTUALIZACIÓN DE ENTORNOS, SINGULARITY.

DESCRIPCIÓN:

En este trabajo el enfoque se basa en como a través de la virtualización de entornos y el paradigma de la computación en la nube, se logra ofertar servicios de visualización remota de alto rendimiento. Se pretende explorar las diferentes formas de como la tecnología de contenedores se puede implementar y beneficiar entornos de computación de alto rendimiento.

La solución propuesta, una plataforma en su versión 2, llamada **YAJÉ-2**, permite a los investigadores acceder remotamente a software de visualización a través de una interfaz web html5; el cual esta implementado sobre el uso de contenedores Linux como parte de una infraestructura HPC. Se propone el uso de contenedores como gestores de entornos virtuales para proveer microservicios, así suplir la demanda de poder de procesamiento y renderizado de usuarios que no cuentan con el software y/o hardware necesario para correr simulaciones a gran escala. Como prueba piloto se simuló y visualizo un caso hidrodinámico perteneciente al área de dinámica fluidos computacionales.

---

[1] Trabajo de Grado en la modalidad de investigación.

[2] Facultad de Ingenierías Físico-Mecánicas, Escuela de Ingeniería de Sistemas e Informática. Director PhD Carlos Jaime Barrios, Codirector PhD Jorge Luis Chacón.

ABSTRACT

TITLE: DISPLAY OF FLUIDS THROUGH TECHNIQUES IN DYNAMICS OF COMPUTATIONAL FLUIDS ON MASSIVE PARALLEL MACHINES BASED ON GPUs.[3]

AUTHOR: CESAR ALEXANDER BERNAL DIAZ[4]

KEYWORDS: CLUSTER, HPC, GPU, LINUX CONTAINERS, SCIENTIFIC VISUALIZATION, ENVIRONMENT VIRTUALIZATION, SINGULARITY.

DESCRIPTION:

In this work the approach is based on how through the virtualization of environments and the paradigm of cloud computing, it is possible to offer high performance remote viewing services. The aim is to explore the different ways in which container technology can be implemented and benefit high-performance computing environments.

The proposed solution, a software platform called **YAJÉ-2**, allows researchers to remotely access visualization software through a html5 web interface; which is implemented on the use of Linux containers as part of an HPC infrastructure. It is proposed the use of containers as virtual environment managers to provide micro services, to meet the demand for processing power and rendering of users who do not have the software and / or hardware necessary to run large-scale simulations. As a pilot test, a hydrodynamic case belonging to the area of computational fluid dynamics was simulated and visualized.

[3] Undergraduate final Project. Research modality.

[4] Faculty of Physical-Mechanical Engineering, School of Engineering and Computer Systems. Advisor: PhD Carlos Jaime Barrios, Co-advisor PhD Jorge Luis Chacón.

Introducción

La Universidad Industrial de Santander con su infraestructura GridUIS-2 (Forero, Uribe, Orostegui, & Mantilla, 2011) integra diversos recursos tecnológicos, entre ellos un clúster de computadoras de alto rendimiento denominado GUANE-1 (Rojas, Vasquez, & Suarez, 2014) localizado en la sede de Guatiguara, ofreciendo así servicios HPC para computo avanzado dirigido a la comunidad científica. Sin embargo, surgió la necesidad de entregar un servicio complementario al procesamiento de alto rendimiento, el cual se denomina visualización remota de alto rendimiento; con el cual se pretende utilizar los recursos disponibles en el clúster GUANE-1, y dirigirlos hacia rutinas de renderizado.

Típicamente los recursos de computo en un Clúster de alto rendimiento se implementan a través de un único Sistema Operativo base de elección y cada nodo a través de rutinas de procesamiento en paralelo orquestadas por un software manejador de recursos, ejecuta trabajos de cómputo para entregar resultados a un nodo maestro; estos resultados (datos) deben ser enviados a través de una conexión de internet hacia el computador del usuario para su posterior visualización e interpretación. Este enfoque se presenta como un "cuello de botella" dado que la cantidad de datos que se necesita transmitir muchas veces está en el orden de los cientos de Gigabytes y por tanto se requiere de gran ancho de banda por parte del usuario que utiliza los recursos HPC.

En adición al tiempo que tarda en procesar una simulación sobre el centro de datos, se requiere que el usuario disponga de unos recursos de hardware mínimos requeridos para poder visualizar datos resultantes, esto a través de herramientas de software que lean estos datos, los interpreten y los representen gráficamente; para este tipo de software muchas veces su utilidad está atada a

licencias que no son costeables por el usuario o el usuario no dispone del hardware necesario para ejecutar dichos programas informáticos.

En este trabajo se presenta una plataforma de visualización que utiliza recursos del clúster GUANE-1, para ofertar un servicio de aplicaciones científicas de forma remota, que permita al usuario que interactúa con esta arquitectura, procesar simulaciones y visualizarlas dentro de la misma infraestructura HPC; un servicio de este tipo pretende dar solución a necesidades de acceso a software que requiere gran potencia computacional e integrar recursos de hardware en una interfaz web para el acceso a la comunidad UIS.

En esta investigación presenta como uno de sus pilares, un proyecto de maestría en ingeniería de Sistemas, YAJE-1, el cual presenta lineamientos similares a los definidos en este proyecto; se busca afrontar y superar las limitaciones de concepto y arquitectura, proponer un nuevo enfoque de desarrollo e integración como metodología de trabajo en el área de ingeniería de software enfocándola a la computación de alto rendimiento sobre arquitecturas heterogéneas basadas en GPU.

En el cuerpo de este documento se propone implementar tecnologías que se están utilizando actualmente en centros de cómputo avanzado y a gran escala; como elemento central para el desarrollo de la plataforma, se propone la tecnología de contenedores, la cual se suma a otras soluciones de virtualización como virtualización de sistemas operativos, y Paravirtualización; dependiendo el escenario de implementación ofrece un rendimiento similar a una maquina real y con mayor flexibilidad y portabilidad que las máquinas virtuales; esto hace viable implementar la tecnología de contenedores Linux sobre recursos de súper computo.

Para el cumplimiento de los objetivos planteados se ejecuta una serie de pasos descritos a continuación que facilitan la comprensión del contenido de este proyecto. Se inicia dando una

breve introducción a conceptos teóricos para el desarrollo de soluciones informáticas de visualización remota y discernir elementos necesarios para la comprensión del caso de prueba de la plataforma como lo es, la visualización de una simulación del área de dinámica de fluidos computacionales.

Acto seguido se discute los resultados de la investigación que precede a este documento, haciendo hincapié en las limitaciones que presenta y una propuesta metodología para entregar un servició viable, mantenerle y escalable.

Se procede a identificar herramientas de software que serán utilizadas en el transcurso de este libro, así como los elementos estructurales de la arquitectura propuesta; Seguido de una detallada selección y comparación de dichas herramientas con el objetivo de mantener la usabilidad de forma indefinida sin requerir licenciamientos que generan un costo agregado.

Finalizada la etapa anterior, se plasma el diseño de la plataforma de visualización remota a través de una serie de figuras exponiendo como se correlaciona cada elemento de software que intervente en la arquitectura.

Se presenta el prototipo desarrollado, las interfaces gráficas, diagramas de integración de la herramienta desarrollada con la plataforma existente clúster GUANE-1, así como la introducción a dos documentos guía, de usuario y de programador, los cuales se dejan como soporte de la plataforma. Y por último se presentan métricas de rendimiento, algunas consideraciones por parte del autor en la sección de discusión, recomendaciones y limitaciones.

1. Objetivos

1.1 Objetivo General

Visualización de fluidos utilizando las ventajas arquitecturales de infraestructuras HPC en máquinas masivamente paralelas basadas en GPUs.

1.2 Objetivos Específicos

- Desarrollar una plataforma prototipo de visualización remota.
- Implementar la visualización de una simulación de un modelo flujo turbulento alrededor de un alabe de una turbo máquina.
- Crear e implementar una estrategia de acoplamiento de la aplicación con una arquitectura HPC disponible basada en GPU.
- Comparar los resultados obtenidos y validar la estrategia seleccionada.

2. Marco de Referencia

2.1 Marco Teórico.

Este trabajo de investigación se enfatiza en la creación y puesta en marcha de un servicio de visualización, adaptando una serie de tecnologías para el acceso remoto a servicios de súper computo, manteniendo una filosofía de código libre, dado las limitaciones de licenciamientos que ofrecen implementaciones privativas.

En este contexto suplir los requerimientos de hardware para la comunidad científica de forma centralizada en arquitecturas tipo clúster, requiere introducir algunos conceptos teóricos como elementos temáticos que permitan cohesionar las tecnologías implementadas y dimensionar el alcance del proyecto, tomando en cuenta conceptos de visualización científica, virtualización, arquitecturas de servicios en la nube, modelo de implementación de la tecnología openGL y una introducción a conceptos del caso de prueba en el área de dinámica de fluidos computacionales, caracterizando únicamente la fase de visualización en un proceso de análisis CFD.

2.1.1 Dinámica de Fluidos Computacionales. CFD (por sus siglas en inglés) corresponde a una rama de la mecánica de fluidos y tiene por objeto estudiar el comportamiento de los fluidos, las interacciones con el medio y sus parámetros como temperatura, presión, velocidad, entre otros; mediante modelos matemáticos y técnicas numéricas por computadora, que confluyen en simulaciones para posteriormente visualizarlas y validarlas buscando la mejor aproximación a las leyes físicas que los rigen. (Mott, 1996)

En el análisis CFD mediante herramientas computacionales, se elimina la complejidad en el análisis de un flujo por el método analítico, siendo esto conveniente en el proceso de diseño industrial, permitiendo así disminuir la necesidad de crear gran número de prototipos, reduciendo el factor ensayo/error, las latencias y las dilataciones y los costes de la fase de producción. (Bhaskaran & Collins)

La etapa de preprocesamiento es donde un investigador realiza análisis matemático del problema y establece la geometría de interés de estudio. La generación de mallas es un proceso crucial en una simulación, dependiendo de la calidad de la malla, se determina en mayor o menor grado la posibilidad de solucionar el caso simulado y la rapidez computacional requerida en la fase de post procesamiento, la complejidad implícita en esta fase dependerá en gran medida del tipo de simulación y las características físicas y condiciones de contorno tenidas en cuenta durante la fase de preprocesamiento.

Luego del proceso de discretización se confluye en un sistema de ecuaciones algebraicas que representan las leyes físicas que serán tenidas en cuenta en el medio discreto del tiempo, estas ecuaciones son procesadas por computadora; se procede a interpretar el resultado de las ecuaciones por medio de software auxiliares que permitan visualizar de forma interactiva, estos datos en la etapa de post procesamiento para su posterior análisis por parte del investigador. La etapa de visualización requiere tanto o más poder computacional que la etapa de preprocesamiento y/o solución de ecuaciones por medio de la computadora. Figura 1

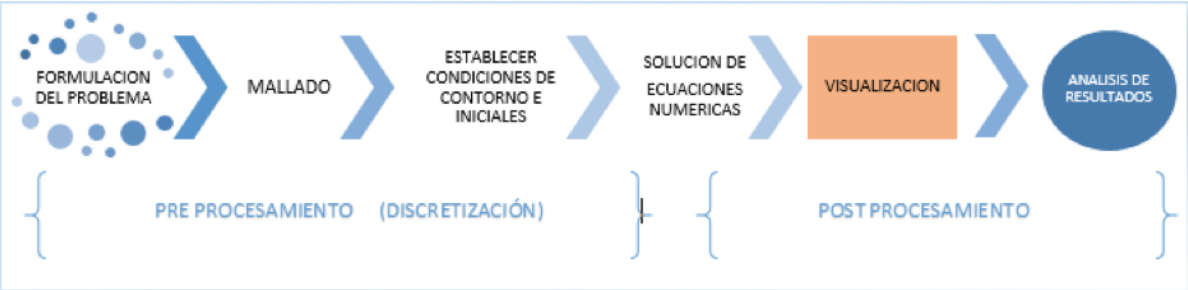

Figura 1 Secuencia de subprocesos en un análisis CFD. Adaptado de Notas sobre

2.1.2 Visualización Científica. El estudio de un fluido es muy importante a la hora de diseñar modelos hidrodinámicos, por esto se ve la necesidad de recrear condiciones de flujo similares (simuladas por computador) a las reales; lo que permite reducir complicaciones mecánicas en prototipos y simplificar la interpretación de resultados.

En un esquema de visualización científica se busca ser rigurosos en las interpretaciones grafica de gran cantidad de datos más que impresionar con efectos visuales, por tal razón utilizando la tecnología de alto rendimiento computacional disponibles para acelerar este proceso, se utilizan metodologías de visualización en paralelo como pasarela en arquitecturas HPC CPU/ GPU. (Parada, 2017)

2.1.3 Virtualización. El termino virtualización se describe como un paradigma de separación de recursos de forma virtual (por software) para suplir una solicitud de servicio de la capa física inferior. Esta metodología se puede aplicar a virtualización de aplicaciones, servidores, almacenamiento y red; y tiene como objeto reducir los costos de implementación de una infraestructura TI, mientras agiliza la fase de despliegue en entornos informáticos robustos. (Riaño, 2017)

2.1.3.1 Virtualización de Sistemas Operativos. Virtualización de SO, también llamada virtualización completa o virtualización con hypervisor, los cuales son aplicaciones que se encargan de simular hardware y presentarlo al software huésped, así mismo son el elemento de control de esta tecnología.

Dependiendo las capas de abstracción implementadas sobre el hardware físico, la virtualización por hypervisor para efectos de esta investigación se clasifica en dos grandes grupos:

2.1.3.1.1 Por hypervisor tipo 1. También llamados nativos o bare-metal, el hypervisor se ejecuta directamente sobre el hardware físico, cargándose a memoria RAM antes que ninguno de los Sistemas Operativos cliente y controla directamente las rutinas de acceso al hardware, enfoque mono usuario.

2.1.3.1.2 Por hypervisors tipo 2. La ejecución del hypervisor se lleva a cabo sobre el contexto de un Sistema Operativo base anfitrión que se carga antes del hypervisor y el Sistema Operativo huésped se ejecuta en un tercer nivel por encima de la capa del hypervisor, enfoque múltiple usuario.

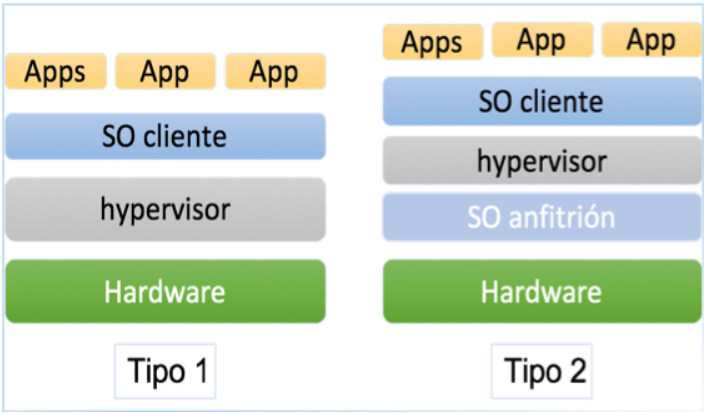

Figura 2 caracterización tecnologías virtualización completa

2.1.3.1.3 Máquina virtual basada en Kernel. Tecnología de virtualización completa basada en el núcleo (kernel Virtual Machine), formada por módulos de núcleo y herramientas de software en espacio de usuario, KVM da a las máquinas virtuales su propio hardware en un entorno virtualizado para trabajar como si fuese sobre maquinas físicas , tiene mejor rendimiento que la virtualización por hypervisor tipo 2 Figura 2 pero requiere soporte de virtualización por hardware por parte del sistema operativo, utiliza sus propios drivers, lo que resta portabilidad entre distintos entornos computacionales.

2.1.3.2 Virtualización de entornos. La virtualización de entornos o también llamada contenedores Linux, tiene como objeto ofrecer un entorno aislado lo más cercano posible a una instalación común de Linux, pero sin la necesidad de tener núcleos separados. Esta metodología facilita mover aplicaciones contenidas entre entornos (desarrollo, pruebas, producción), mientras mantiene completa funcionalidad.

Los contenedores Linux reducen conflictos entre equipos de desarrollo y soporte, los desarrolladores pueden enfocarse en sus aplicaciones y el personal de operaciones se enfoca en la infraestructura, mientras mantienen una comunicación complementaria; además los contenedores al estar implementados sobre tecnologías de software libre proveen siempre versiones de software recientes, compatibles con la mayoría de hardware y logran ser una tecnología ampliamente adaptable a entornos de computación de alto rendimiento. A través de características del sistema operativo Linux como “Namespace” y “Cgroups”, esta tecnología permite aislar procesos y asignar recursos de hardware a cada contenedor. (FIRESMITH, 2017) Figura **3**

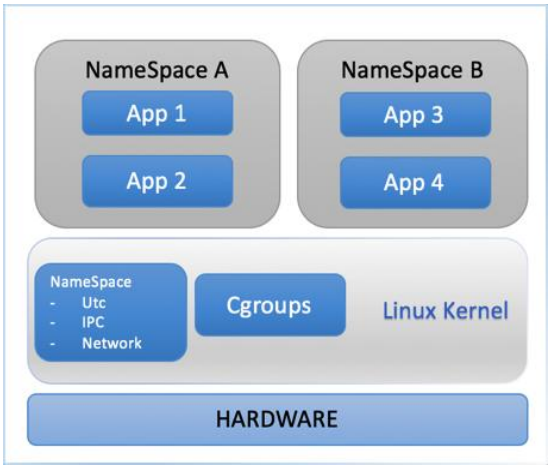

Figura 3 Anatomía de contenedores Linux

2.1.4 Arquitecturas de servicios en la nube. La computación en la Nube es un paradigma para ofertar servicios de computo virtuales a través de internet, el cual se puede clasificar según la gestión de recursos de TI presente.

La computación en la nube tiene como objeto reemplazar las maquinas locales con centros de datos privados o públicos con infraestructura virtual; permite escalar la infraestructura para aprovisionar en tiempo real los requerimientos de un cliente. Estos recursos se presentan como servicio en las siguientes categorías: (Ullah & Xuefeng)

2.1.4.1 Software como servicio (SaaS). Las aplicaciones se diseñas para usuarios finales y se entrega un servicio transparente para el usuario ocultando detalles de la arquitectura subyacente.

A través de este modelo se presenta una amplia variedad de funcionalidades, como clientes de correo electrónico, paquetes de ofimática, hospedaje para video juegos, aplicaciones de diseño gráfico, entre otras más.

2.1.4.2 Plataforma como servicio (PaaS). Tiene como objeto entregar un entorno de desarrollo completo abstrayendo al desarrollador de la necesidad de gestionar la infraestructura, en cuanto a instalación de aplicaciones, actualizaciones, redes, y mantenimiento de la plataforma.

2.1.4.3 Infraestructura como Servicio (IaaS). Corresponde a la forma más elemental de computación en la nube, brinda a los usuarios acceso a elementos infraestructurales TI, tales como espacio de almacenamiento, redes, una API para interactuar con los recursos, bases de datos. Es el modelo más elástico para prestar servicios de computo remoto, pero requiere que el usuario posea fuertes conocimientos informáticos y para el equipo de soporte implica desarrollar una infraestructura solida con principios de seguridad y alta disponibilidad de recursos.

2.1.4.4 Arquitectura Clúster de alto rendimiento. Un clúster es una colección de computadoras interconectadas que trabajan en conjunto para prestar un servicio, distribuyéndose las tareas y para el usuario estas máquinas actúan como una sola.

Para alcance de esta investigación, se implementará un nodo de visualización a un clúster de alto rendimiento; este tipo de clúster se caracteriza optimizar los procesos ejecutados en los diferentes nodos, tratando de obtener el mayor número de operaciones en punto flotante con el menor tiempo posible.

Un clúster consta de por lo menos dos o más nodos, los cuales están conectados entre sí por un canal de comunicación, y necesitan un software de control especializado en distribuir tareas a cada uno de los nodos; la topología de un clúster de alto rendimiento puede presentarse como homogéneos formado por equipos de la misma arquitectura o heterogéneos, integrados por nodos

con distinciones como arquitectura, sistema operativo, rendimiento de CPU o acceso a recursos. Figura [3]

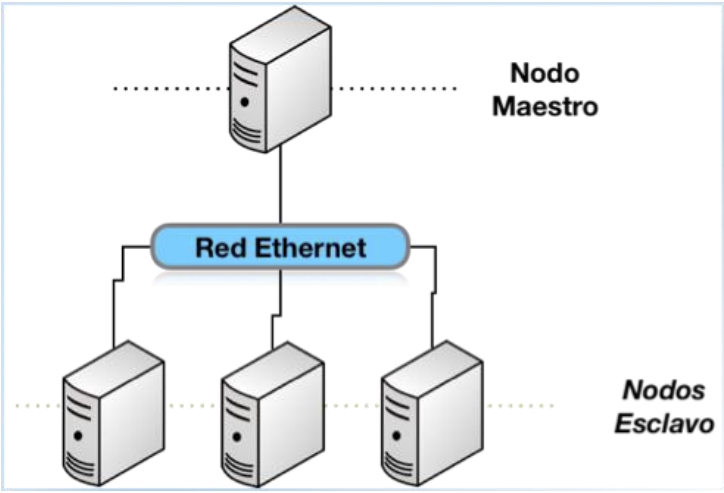

Figura 3 Arquitectura base de un clúster

2.1.5 Arquitectura de implementación OpenGL. OpenGL es una API que permite a las aplicaciones, interactuar con el hardware de video como unidades de procesamiento grafico (GPU) y pantallas; el uso de esta API por su flexibilidad e independencia de la plataforma se ha convertido en el estándar de la programación grafica en entornos Unix y derivados. Si una particular pieza de hardware no puede implementar todas las características de la especificación OpenGL vía hardware (renderización por hardware), los vendedores de hardware deben proveer un mecanismo alternativo que integre esta funcionalidad, típicamente vía software (renderización por software) con el inconveniente de entregar menor rendimiento que el modelo de renderización por hardware. (Webmaster, 2018)

2.1.5.1 Sistema de ventanas X. El sistema de ventanas X es un Framework que provee un mecanismo de interacción con hardware para aplicaciones que posean una GUI, ejemplo de ello, mover una ventana, eventos de ratón, teclado. X se concibió para ser independiente del sistema operativo y provee un servidor grafico X propiamente dicho y un protocolo de comunicación Xorg.

El sistema de renderizado de una aplicación en un sistema heredero del sistema operativo Unix, se basa en un modelo cliente servidor usando varias formas de comunicación de procesos entre la aplicación y el hardware. Si una aplicación necesita usar OpenGL, primero pregunta al servidor X para crear un contexto de trabajo, una vez el contexto está listo, OpenGL puede hablar directamente a la GPU, saltándose la comunicación con el X server. Figura 4

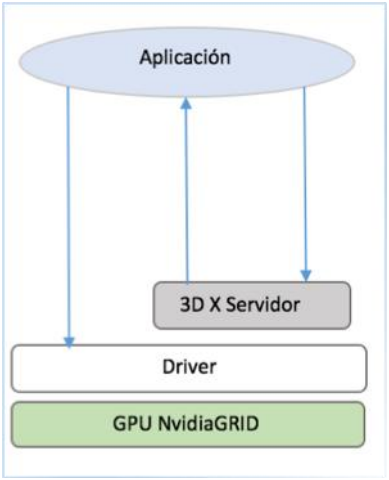

Figura 4 Uso API OpenGL con renderización local. Adaptado de (**Nvidia, 2014)**

2.2 Metodología.

Se propone implementar la metodología en cascada; esta metodología es apropiada, ya que los requerimientos son establecidos al inicio en la fase de investigación preliminar y permanecen invariantes durante el tiempo destinado para la consecución del software.

2.2.1 Investigación preliminar. En esta fase, se recopilan datos e información sobre el estado del arte utilizado para afrontar las necesidades que se quieren ejecutar con la nueva aplicación en entornos HPC como servicio de visualización remota. Es importante definir la selección de recursos tales como herramientas de software, tiempo, recurso humano.

Se concluye esta fase con la especificación de requisitos y la elaboración de la propuesta.

2.2.2 Diseño. La finalidad de esta etapa es lograr el diseño del modelo funcional, estructurándolo a partir de la especificación de requisitos de la fase de investigación preliminar. Adicionalmente se caracterizará la interfaz de la aplicación en la cual se determina el modelo de acceso y reserva de recursos de visualización dentro del clúster GUANE-1.

2.2.3 Prototipado. En esta fase se realiza la codificación de los objetivos de la etapa de diseño; concretando los requerimientos obtenidos en la fase de investigación preliminar junto con los datos, modelos y diagramas generados en la etapa de diseño. Se procederá a la codificación de unidades estructurales del proyecto, validación funcional de las mismas.

En este proyecto se usará los lenguajes Python para facilitar la integración de trabajos de programación futuros, y *batch* para codificar la interacción entre los recursos de software, módulos de validación y control de la asignación de recursos dentro del nodo de visualización.

2.2.4 Implementación. Se interrelacionan las fases anteriores con la implementación sobre el hardware y arquitecturas disponibles. Se formulan las pruebas de integración y procede con el acoplamiento sobre la plataforma distribuida de cálculo científico clúster GUANE-1.

2.2.5 Evaluación y mantenimiento. Se procederá con el estudio del caso de prueba y la visualización remota, buscando en cada aspecto retroalimentación para mejorar la interacción de la plataforma con el usuario y/o investigador.

Se propone implementar una metodología vanguardista que interrelacione a desarrolladores y personal de operaciones para generar un producto de calidad, denominada DevOps.

Se realizan correcciones de diseño y de programación mediante un modelo evolutivo; buscando un prototipo funcional y disponible para despliegue como plataforma como servicio de visualización científica para la comunidad universitaria UIS.

Se procede a plasmar información relevante al proyecto de investigación en un documento final en pro de la sustentación del trabajo de grado para generar un aporte significativo a los grupos de investigación científica que lo requieran.

3. Lineamientos.

En esta sección se pretende abordar una investigación precedente a este proyecto, la cual fue tomado como punto de referencia, analizando las limitaciones presentadas, ventajas y desventajas existentes. Así mismo presentar los lineamientos y alcances de esta investigación.

3.1 Discusión de Antecedentes

Mediante el proyecto de maestría denominado Computación de alto rendimiento para la visualización e interacción remota de aplicaciones científicas, se definió como objetivo, diseñar un esquema de procesamiento, visualización de datos y colaboración con aplicaciones científicas. Este enfoque se propuso esquematizar y modelar una arquitectura de visualización remota para entornos HPC, partiendo de los lineamientos funcionales del clúster GUANE-1, y entregar una plataforma de nombre YAJE-1.

Ventajas:

- Está enfocado a solventar los requerimientos de renderizado remoto, a partir de soluciones informáticas ampliamente utilizados en la industria, tales como soluciones de máquinas virtuales para entornos corporativos.
- Propone una integración a la infraestructura GUANE-1,
- La solución presenta mecanismos para garantizar la fiabilidad de la información representada, debido a se soporta sobre soluciones de software con amplio soporte técnico.

Desventajas:

- La solución propuesta, limita su alcance a soluciones de virtualización privativos, con requerimientos de licencias de pago.

- El modelo de acceso a la plataforma está definido a partir de las implementaciones existentes del software VMware, requiere un conocimiento previo por parte del usuario para desplegar entornos virtualizados.
- Presenta limitaciones de control de recursos de hardware, y tiempo de uso de los mismos, debido a que la solución propuesta, no se integra con el software existente SLURM, el cual está encargado de gestionar los recursos informáticos del clúster.
- Se requiere desplegar una serie de nodos adicionales de carácter administrativo, tales como *vCenter, connection Server* y *security server*; los cuales limitan el modelo de acceso a la infraestructura y genera una carga de trabajo extra al personal de soporte.
- El esquema de vGPU implementado, limita la cantidad de usuarios que requieren procesamiento de alto rendimiento sobre la plataforma.
- Desplegar un entorno de escritorio completo por usuario de forma remota, consume más ciclos de CPU, memoria RAM y ancho de banda, comparado a soluciones de acceso aislado a aplicaciones.

3.2 Alcances del proyecto

Luego de enfatizar los requerimientos funcionales a partir de la discusión de antecedentes,

Se tiene como propuesta entregar una plataforma con las siguientes características:

- Implementar una plataforma de visualización que carezca de limitaciones en estudios anteriores.
- Utilizar tecnologías que se estén implementado como estado del arte.
- Utilizar únicamente software libre.

- Especificar elementos, mecanismos y metodologías de integración con recursos disponibles
- Diseñar e implementar pruebas de funcionalidad de la plataforma con un caso real como prueba verídica de las limitaciones de la implementación.
- Implementar bases de conceptos de ingeniería de software al área de super computo, para estandarizar modelos de trabajo en búsqueda de reproducibilidad.
- Crear una solución que requiera la menor cantidad de conocimientos previos para el usuario.
- Implementar una solución de software que implemente conceptos como computación en la nube, metodologías agiles, plataforma como servicio y así generar una línea de desarrollo más robusta.
- diseñar un modelo de acceso de cero instalaciones en el lado del cliente.
- Generar lineamientos a partir de guías para el usuario y de programador con el fin de facilitar la usabilidad e integración de la herramienta a cualquier entorno HPC.
- Crear un servicio de carácter competitivo a otras soluciones, discernir el estado del arte en cuanto a otras metodologías existentes.

4 Desarrollo de la plataforma de visualización remota

El desarrollo de la plataforma se ha dividido en selección de herramientas de software, seguido de modelado del nodo de visualización

4.1 Selección de herramientas de software.

La selección del software a utilizar, paso a través de varios filtros, se especifica por qué se eligió cada uno de ellos para la implementación de la plataforma a través de comparaciones de las distintas tecnologías.

4.1.1 Tecnología de virtualización. Teniendo en cuenta que el portafolio de servicios de un centro de supercomputación es muy extenso, la cantidad de paquetes de software dentro del mismo sistema operativo se torna complejo de mantener y puede generar incompatibilidades entre las distintas configuraciones, se ha decidido implementar un modelo de aislamiento de software.

Se diseñó una tabla comparativa para seleccionar la tecnología de virtualización más apta para implementar en un entorno HPC y prestar un servicio de código abierto de visualización remota, se tomó en cuenta como factores prioritarios, rendimiento, tiempo de despliegue, costos, utilización de recursos. Tabla 1

Tabla 1 *Comparación Máquinas virtuales y contenedores Linux*

Maquinas virtuales	Contenedores
VENTAJAS	
Portabilidad entre distintos sistemas operativos	Flexibilidad y portabilidad entornos Linux
Aislamiento total de la plataforma subyacente	Simplifica la entrega de servicios en entornos Linux
Tecnóloga soportada por gran cantidad de empresas	Integra al equipo de trabajo desarrolladores, investigadores y soporte (filosofía DevOps)
Administración remota de fácil acceso	Se pueden desplegar gran cantidad de contenedores en cuestión de segundos
Posee mediana curva de aprendizaje	Baja utilización de recursos del sistema.
Plataformas estables y menor tiempo de recuperación en caso de fallas.	Los costos de implementación son menores que las maquinas virtuales.
Ampliamente adaptable para las políticas empresariales.	Facilita la integración en proyectos de software complejos a través de API's
Facilidad de configuración	Retorno de inversión a corto plazo
Presenta gran variedad de implementaciones según los requerimientos	Permite que los equipos humanos de tecnología sean pequeños
Facilidad de incorporación de periféricos hacia la maquina virtual, discos en red, pantallas anidadas, etc.	Garantizan coherencia durante todo el ciclo de desarrollo de un software (Estandarización)
DESVENTAJAS	
Ocupan gran cantidad de recursos del sistema.	Requiere reforzar la seguridad de cada contenedor desplegado
Requiere una copia del sistema operativo por cada unidad maquina virtual.	tecnología en desarrollo, soportado por pocas empresas.
En producción requiere licencias costosas.	Solo esta soportado en sistemas operativos que usen Linux como núcleo.
Necesita computadores de excelentes prestaciones técnicas para entregar servicios complejos.	La separación de recursos del sistema debe ser manejada cuidadosamente para evitar que el contenedor consuma todos los recursos del nodo.
Aísla los grupos de trabajo, desarrolladores y soporte.	Agrega complejidad de trabajo al personal de soporte.
El computador que hace funcionar la maquina virtual es un componente critico de la arquitectura.	No instancia su propio kernel, por tanto no es posible instalar módulos de kernel (controladores) para entornos de desarrollo y soporte de nuevo hardware en caliente.
Requiere mucho tiempo para el arranque de la maquina virtual.	Requiere Alta curva de aprendizaje previo.

Como métrica complementaria de rendimiento, al comparar máquinas virtuales con contenedores Linux, se tuvo en cuenta una investigación de terceros, donde realizan benchamark Intel Linpack sobre tecnologías de virtualización en entornos HPC aplicado a la dinámica de fluidos computacionales. Comparación virtualización con efectos de red. Figura 5. Comparación unidades de punto flotante en un mismo host, Figura 6

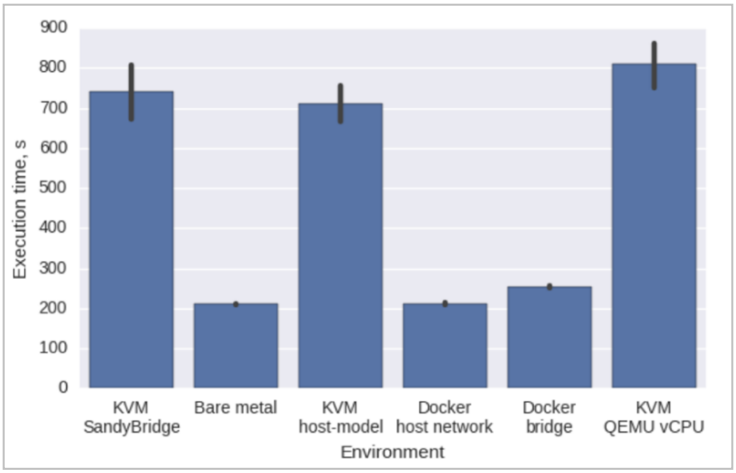

Figura 5 Comparación del tiempo de ejecución de un solver CFD, fuente (**Ermakov & Vasyukov, 2017)**

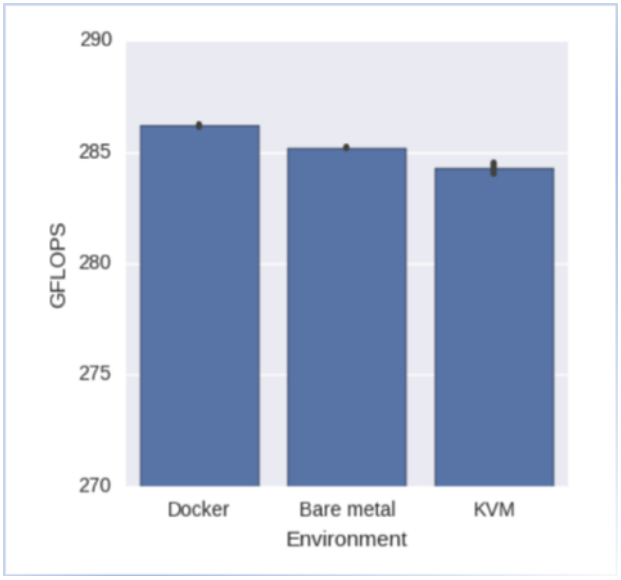

Figura 6 Gflops contra la tecnología de Docker. Fuente **(Ermakov & Vasyukov, Testing Docker Performance for HPC Applications, 2017)**

Acorde a la información anterior recolectada se comprobó que los Contenedores presenta similar rendimiento a las tecnologías más representativas de virtualización de sistemas operativos, bare-metal (Vmware, 2006) y KVM (Kvm Project, 2018) y en promedio el doble de rendimiento en cuanto a tiempo de despliegue; también los contenedores son más aptos para el desarrollo de aplicaciones como microservicios, lo cual es más coherente con los servicios que presenta un clúster de alto rendimiento.

4.1.2 Tecnología grafica de acceso remoto. Las tecnologías de acceso remoto tienen como objetivo ofertar servicios con un modelo independiente de la plataforma, por tanto, se requiere de una comparación de estas herramientas y los diversos protocolos de comunicación para seleccionar la tecnología más apropiada de acceso remoto en un entorno HPC.

Se realizó una valoración de las diversas tecnologías existentes que permiten la visualización remota, filtrando información para adaptarla a las necesidades de la plataforma desarrollada. Tabla 2

Tabla 2 *Comparación de las tecnologías de acceso remoto*

Tecnologias de acceso remoto	Protocolo	Licencia	Carga Util	Ventajas	Desventajas
vnc	RFB	opensource	pixeles	independiente de la plataforma, compresion inerente al protocolo	fuerte dependencia del servidor X 3D base, y compleja configuración de seguridad
terminal services	RDP	privativo	primitivas graficas	bajo consumo de red	dificultad de mantener fuera de entornos windows
X11 Forwarding	ssh	opensource	primitivas graficas	compresion de datos a traves de ssh	consumo de ancho de banda elevado
Citrix XenApp	ICA	privativo	pixeles	multiplataforma, soporte	debil implementacion multiusiario - multi escritorio
Tecnologias de acceso remoto	**Costo**	**Complegidad de implementación**	**Soporte Web**	**Enfoque**	**Versiones**
vnc	gratuito	baja	SI	escritorio virtual	cliente y servidor
terminal services	gratuito limitado/ de pago	media	NO	acceso remoto real, multiusario	cliente y servidor
X11 Forwarding	gratuito	baja	NO	tranfiere X comandos a un servidor remoto	cliente y servidor Linux
Citrix XenApp	de pago	alta	SI	entornos SaaS, cloud, desktop	cliente y servidor

En base a que el modelo cliente-servidor de una aplicación que requiere openGL Figura 7(simplificada local), existe una metodología que permite que una aplicación que se ejecuta en un equipo remoto sea representada gráficamente de forma remota, colocando el servidor X (Packard) del lado del cliente y transfiriéndole primitivas graficas por la red e instrucciones de renderizado.

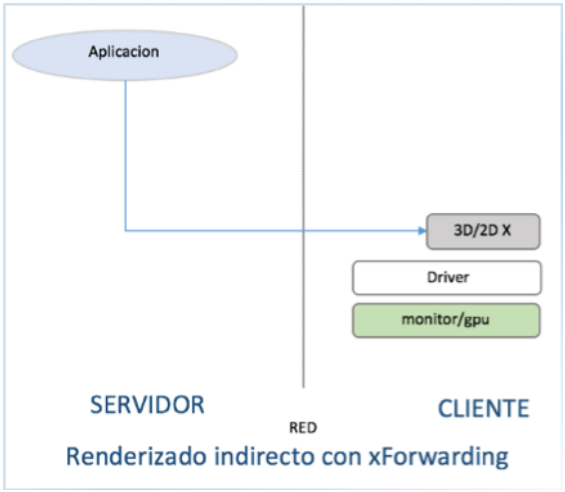

Figura 7 Tecnología de renderizado xForwarding

Debido a que uno de los objetivos del desarrollo de la plataforma de visualización es aprovechar la de potencia de cómputo y representación gráfica remota del lado del servidor, se descarta la tecnología xForwarding por las siguientes desventajas:

- Aunque la aplicación se ejecuta del lado del servidor, se requiere que el cliente posea gran poder computacional para representar gráficamente la información enviada por la aplicación desde el servidor.
- El protocolo X es muy detallado y agrega sobre carga en la información que envía en una conexión; en redes de alta latencia o bajo ancho de banda las conexiones no son realmente buenas.

- Es necesario proveer software servidor X en el lado del cliente. Ejemplo: Windows Exceed (Center, 2009) (comercial), Cygwin/x (Cygwin Corporation, 2018) (Open Source).

la tecnología de visualización remota elegida fue **VNC** (Richardson, Stafford-Fraser, Wood, & Hopper, 1998), por sus características como modelo cliente/servidor, soporte web, independencia de la plataforma y tipo de licencia.

4.1.3 Software servidor. Luego de seleccionar la tecnología grafica de acceso remoto más apta para el tipo de servicio que se desea ofertar, es inherente elegir el software que se encargara de procesar y enviar datos al cliente remoto. Tabla 3

Tabla 3 *Comparativa software servidor.*

software	Protocolo	Licencia	Ultima version estable	3D acceleration por hardware - servidor	SO (server version)	CIFRADO	Multi- sesión
Chicken of the VNC	RFB(VNC)	*GPL*	2011-02- 2.1.1	NO	Si	SSL	NO
QVD	Nx - http	*GPL*	2017-00-4.0	NO	SI	SSL	SI
Remmina	RDP, RFB, SPICE, XDMCP, SSH	*GPL*	2017, 1.2.0	NO	NO	AES 256	SI
ssh X forwarding	X11	*BSD*	2015, openSSH 7.1	No	SI	VARIOS	SI
TigerVNC	RFB(VNC)	*GPL*	2017, 1.0.8	NO	SI	SSL, TLS	SI
turboVNC	**RFB(VNC)**	***GPL***	**2017, 2.1.1**	**SI**	**SI**	**SSL, TLS, SSH**	**SI**
Xpra	yaml, RFB(VNC)	*GPL*	2017-12-20-2.2.1	NO	SI	SSH, TLS, AES	SI
X2Go	Nx - http	*GPL*	2017, 4.1	NO	SI	SSH	SI

El software turboVNC fue elegido como servidor VNC porque es un proyecto OpenSource agnóstico a virtualGL el cual fue desarrollado paralelamente con los mismos objetivos, la computación grafica de alto rendimiento; adicional implementado a la compresión de datos y

soporta múltiples sesiones virtuales, para un entorno multiusuario apto para el modelo de servicios que oferta el Clúster GUANE-1 de la UIS.

4.1.4 Software Cliente. Para diseñar una plataforma donde no se requiera instalar un paquete de software en el cliente, se optó por una perspectiva de aplicaciones embebida en un entorno web; desde esta perspectiva se comparó software cliente, que se logre ejecutar sobre un servidor web y permita integrar las demás tecnologías elegidas.

Dos del software principal de código libre activos, que dan soporte a escritorios virtuales desde entornos web son noVNC (Martin, s.f.) y el segundo de nombre Apache Guacamole (The Apache Software Foundation, 2018), se comparan mediante la Tabla 4.

Tabla 4 *Comparación de los clientes web vnc noVNC y Guacamole*

NoVNC	Guacamole
Orientado a entornos IaaS, PaaS	Plataforma robusta que agrega sobre carga a la comunicación entre procesos.
codificación / decodificación de datos incorporada	Orientado al servicio de video remoto.
acceso directo a la sesión VNC sin configuraciones extensas.	agnóstico al protocolo RDP

Debido a que Guacomole, tiene su funcionamiento sobre el protocolo RDP, se optó por noVNC como cliente vnc web.

4.1.5 Tecnología de bifurcación openGL. Para prestar un servicio de visualización de alto rendimiento, es necesario implementar un mecanismo que garantice el que el renderizado 3D se ejecute en el lado del servidor, buscando eficiencia en la administración de recursos; en este aspecto algunas herramientas en el mercado implementan una metodología para interceptar la comunicación con el hardware gráfico y permiten disminuir la cantidad de información que es enviada al cliente remoto.

Existen dos proyectos de código libre que agregan soporte OpenGL a clientes remotos, VirtualGL (VirtualGL Project, 2018) y Primus (amonakov, 2015)

La elección de VirtualGL sobre Primus se debió a que este último está limitado a soporte de arquitecturas Nvidia Optimus y no es un proyecto activo a la fecha de realización de esta investigación.

4.1.6 Estado del arte Acorde a la estrategia seleccionada se propone comparar la herramienta desarrollada con las tecnologías que se están usando en algunos centros de investigación y entidades públicas, esto con el fin de identificar los principales competidores en el área de visualización remota HPC. Tabla 5

Tabla 5 *Comparativa proyectos de visualización remota por institución*

Institución	Proyecto	Pais	tecnologia	Plataforma	Entorno	intervención del usuario
CINECA [30]	RMC	ITA	VNC	Maquina virtual	Cliente servidor - web	medio
Universidad de Arizona [31]	TACC	USA	Vnc + virtualGL	Docker	Cliente servidor	medio
Nvidia [32]	viztesla	USA	NICE solución	Docker - lxc - singularity	web	bajo
Universidad de Standford [33]	remoteViz	USA	Nx technology noMachine	Maquina virtual- bare-metal	Cliente-servidor	alto
Nasa Centro de simulación Clima [34]	uv-cdat	USA	Turbovnc + virtualGL	Maquinas virtuales	web	medio
Universidad de Oxford [35]	remoteService	UK	Vnc + noMachine	Bare-metal , docker	Cliente servidor	alto
Universidad de Yale [36]	No name	USA	X-forwarding - vnc	Bare-metal- maquina virtual	consola	alto
NICE [37]	DCV	ITA	rdp + hardware compression	Maquinas virtuales	Cliente servidor - web	medio
TCBG [38]	remoteviz	USA	Nice solución + vnc client	Maquina virtual	consola	alto
UNAM [39]	no-name	MEX	vnc	bare-metal	consola	alto
UIS	YAJÉ	COL	Turbovnc + virtualGL + noVNC	singularity containers	consola-web	medio-bajo

4.2 Descripción de las herramientas seleccionadas.

En este apartado de la selección de herramientas que intervienen en la investigación se sintetiza el modelo de funcionamiento para cada software.

4.3 Componentes del nodo de visualización.

Luego de la selección de herramientas se procedió a integrar las distintas tecnologías y plasmar los mismos, a través de una serie de diagramas, donde se esquematizo las distintas conexiones lógicas.

Como elemento estructural a nivel de software y portabilidad se decidió implementar un contenedor para cada aplicación que sea solicitada como servicio remoto y un contenedor independiente que será el elemento de comunicación de la aplicación con el entorno web y se encargara de mantener disponible la sesión de usuario, aunque el cliente se desconecte para cambiar de terminal, en otras palabras, este modelo se pensó con persistencia de sesión facilitando

la reconexión en caso de fallas de red. El enrutamiento de las sesiones VNC se definió dentro del contenedor de aplicaciones por cada cliente conectado, de esta forma se refuerza el modelo de seguridad independizando las sesiones web en un entorno controlado limitado. Figura [8]

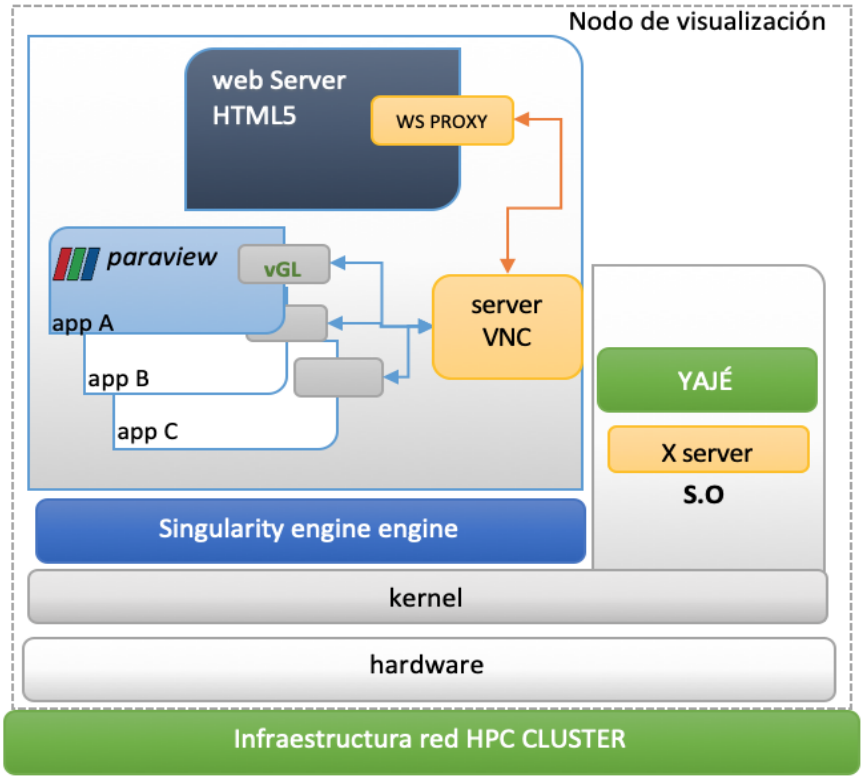

Figura 8 Estructura organizativa elementos estructurales nodo de visualización

4.3.1 Modelo funcional del nodo de visualización. La arquitectura de la plataforma desarrollada se subdividió por componentes de software, indicando el flujo de información entre bibliotecas compartidas, ejecutables y paquetes de software. Del mismo modo con el objetivo de estandarizar conceptos involucrados, se indica las distintas tecnologías que intervienen en la comunicación mediante un modelo abstracto. Figura [9]

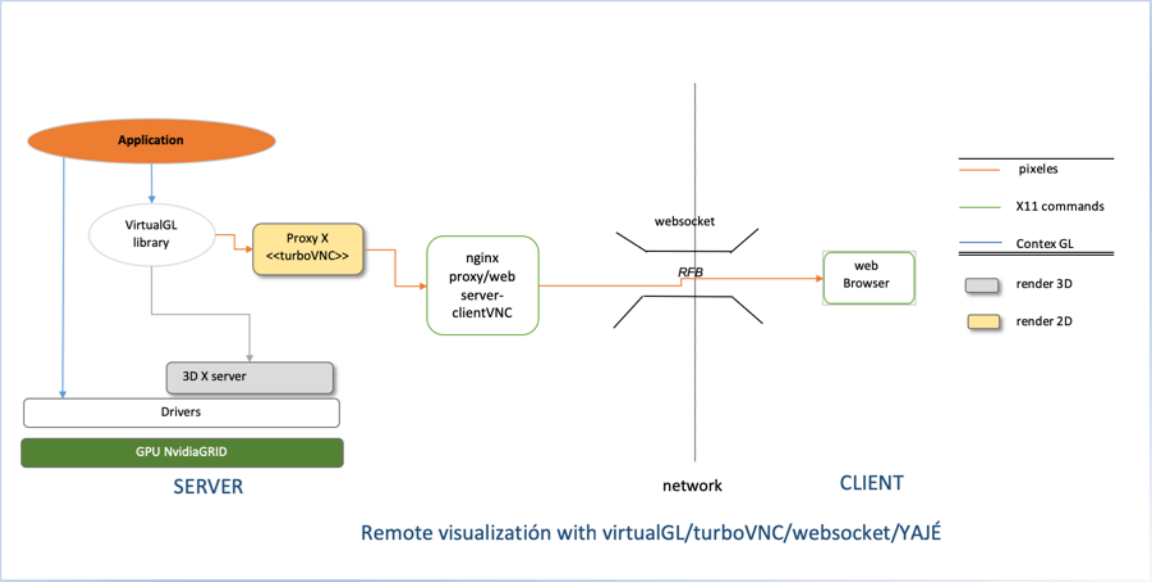

Figura 9 Esquema de comunicación por software para la plataforma de visualización

Este modelo abstracto de la plataforma está basado en la implementación de noVNC para la plataforma de visualización remota YAJÉ, representa el flujo de información gráfica desde el servidor hacia el cliente, y se indica los flujos de Renderizado 2D y 3D y en que segmento de la arquitectura se procesan.

En este contexto la información transferida al cliente corresponde a Imágenes portadas por el protocolo RFB, que se envían a través de un canal websocket/ssh de comunicación full-duplex, desde el servidor web hasta el software cliente web.

4.3.2 Modulo de acceso al nodo. El modelo de acceso al nodo de visualización será dispuesto a través de una sesión de re direccionamiento de puertos SSH cifrada, como esquema lógico y buscando facilitar la comprensión del cliente al momento de realizar la conexión web VNC sobre SSH se dispuso una figura que simplifica este mecanismo, enfatizando que aplicar esta metodología, se elimina la necesidad de que el usuario mueva los datos de las simulaciones efectuadas a través de la red, y el renderizado se realice en el mismo centro de cómputo. **Figura 10**

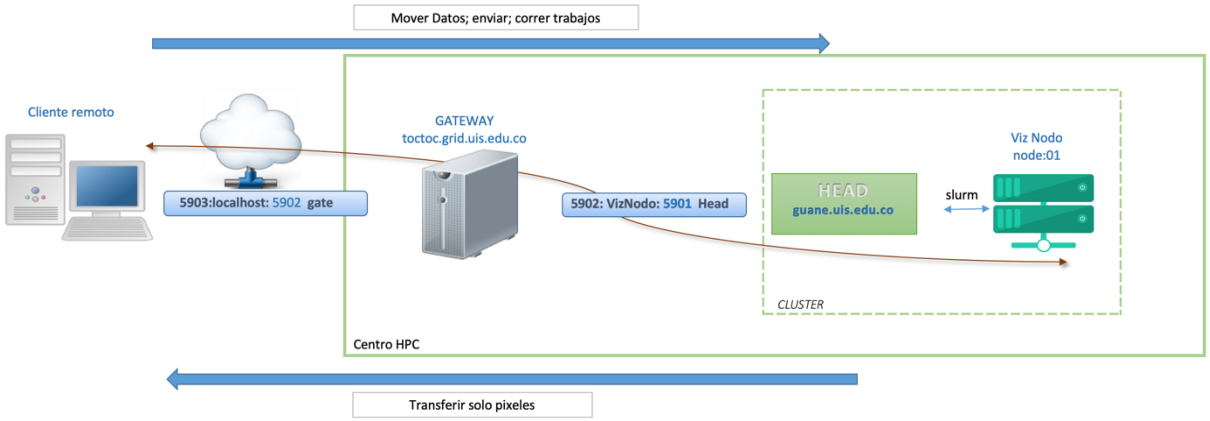

Figura 10 Modelo de acceso al nodo de visualización

5. Visualización caso hidrodinámico.

En esta sección tiene por objeto demostrar las ventajas ofrecidas al investigador que utiliza la plataforma desarrollada, a través de la visualización de un caso real en el área de dinámica de fluidos computacionales sobre la arquitectura desarrollada.

El caso de prueba utiliza el software Paraview (Kitware, 2018) para renderizar datos generados en un proceso previo de simulación computacional sobre el clúster GUANE-1, este caso corresponde a una simulación CFD en el software OpenFOAM (ESI Group, 2018), para el diseño experimental de un convertidor de energía de olas, del tipo columna de agua oscilante (OWC), totalmente sumergido en un canal de oleaje.

Uno de los objetivos de ParaView es permitir el análisis de grupos de datos muy grandes, de orden de magnitud de peta bytes; este software tiene mejor rendimiento sobre recursos de súper computación que sobre máquinas de escritorio limitadas; por tal motivo el uso herramienta se convierte en un método que tiene el potencial de medir el rendimiento de la plataforma.

Se cuenta con los datos experimentales, los cuales permiten validar el modelado CFD, que replica numéricamente este modelo físico, y así, estudiar su comportamiento hidrodinámico, con el fin de determinar la influencia geométrica y encontrar mejores condiciones de funcionamiento. Tabla 6

Tabla 6 *Datos resultantes de la simulación*

caracteristica	valor
largo [m]	11
alto [m]	1.2
ancho [m]	0.4
delta simulación [s]	0.001
celdas	11,025,486
puntos	11,257,921
tamaño en disco [GB]	543
tipo	multibloque
Tiempo de simulación [s]	17

Debido a que los requerimientos para representar los datos de la simulación en RAM para el caso hidrodinámico [Tabla 6] supera ampliamente a las especificaciones de hardware en el nodo de visualización, se optó limitar el proceso de renderizado a un segundo de simulación Tabla 7

Tabla 7 *Parámetros prueba de Renderizado caso real.*

caracteristica	valor
tamaño en ram [GB]	7.5
Delta de Tiempo de renderizado [s]	1.3
Tiempo de lectura HDD [s]	30
FPS	467

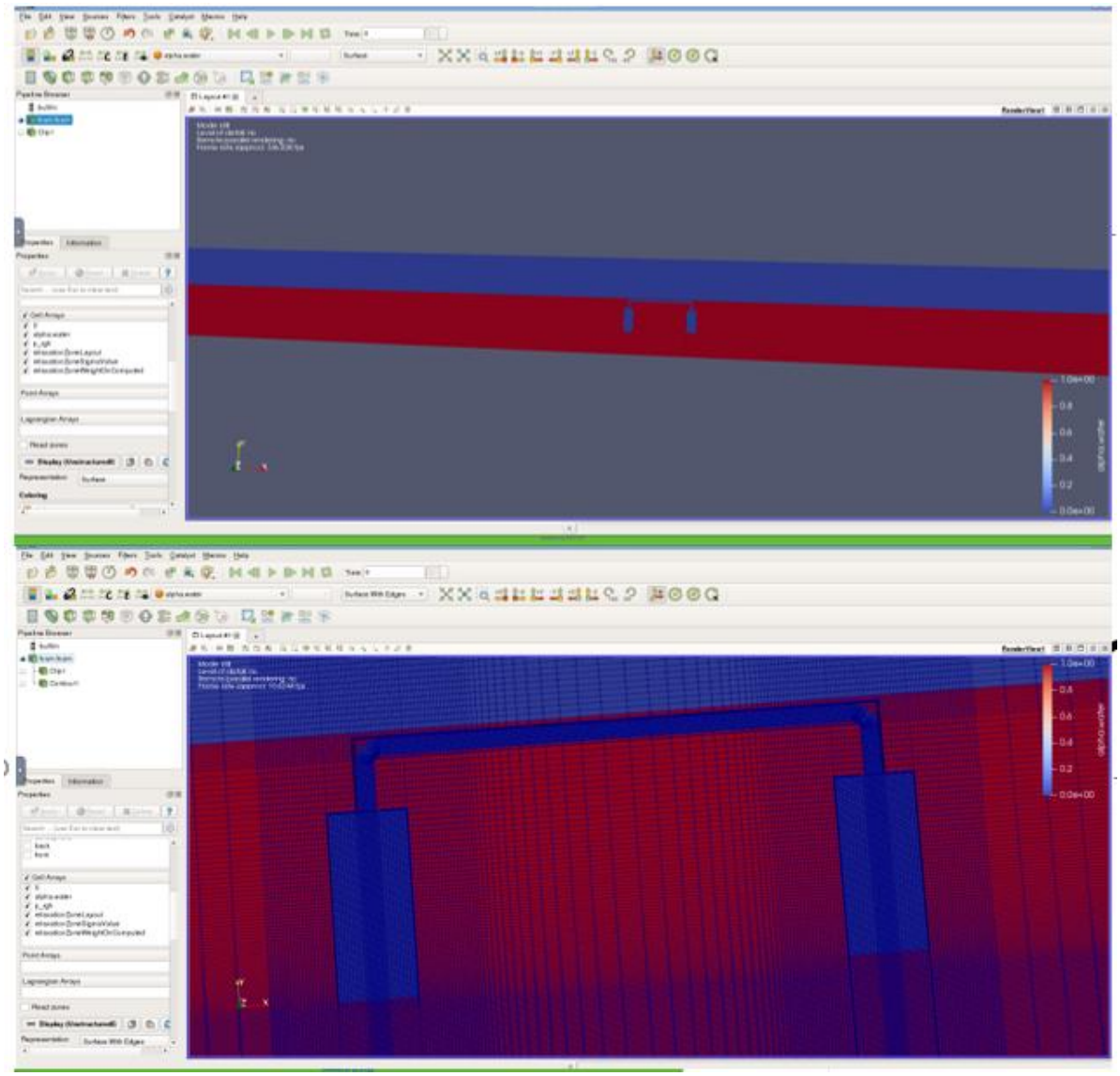

Figura 11 Renderizado en Paraview de un convertidor de energía de olas

Análisis Caso de prueba Figura 13:

Acorde a la información recolectada en la tabla 7, se puede concluir para el caso de prueba.

- Se requiere implementar la plataforma de visualización sobre hardware de mejores prestaciones para generar métricas de rendimiento.

- En redes de baja latencia la aplicación tiene un buen rendimiento, esto se constata con el número elevado de FPS generados en el proceso de renderizado.
- La cantidad de datos leídos en disco representa una limitante al momento de renderizar estos datos independientes de la aplicación utilizada.
- El caso de prueba se presenta como una prueba de estrés de hardware el cual no es posible llevarlo a proceso de renderizado en un computador con recursos limitados; por tanto, la necesidad de renderizado en el centro de cómputo es viable y necesaria.
- Para esta simulación con el software Paraview, no se tuvo en cuenta la capacidad de este para renderizado en paralelo, debido a que la implementación de la plataforma consta de un solo nodo.
- Se logro acortar el tiempo de toma de decisiones del investigador para este caso de prueba debido a que el modelo de renderizado in-situ al centro de cómputo, elimina el tiempo de espera al eliminar la necesidad descargar los datos de la simulación en el equipo de cómputo del usuario.

6. Estrategia de acoplamiento sobre una arquitectura HPC.

Debido a que todo servicio de visualización representa una alta carga de procesamiento y lectura de datos, para limitar los cuellos de botella en la lectura y escritura de datos, se propuso integrar el nodo de visualización a la red existente en el clúster correspondiente a la red infiniband y así se mantiene invariante la topología de red. (SC3 UIS, 2017)

La Figura 12 Topología de red de la infraestructura GUANE-1expone de manera más clara la distribución y modelos de acceso al clúster integrando el nuevo servicio de visualización remota, se diferencian dos tipos de red, la red de 10 Gbps indicada en azul y la red infiniband de acceso rápido para cargas pesadas de trabajo.

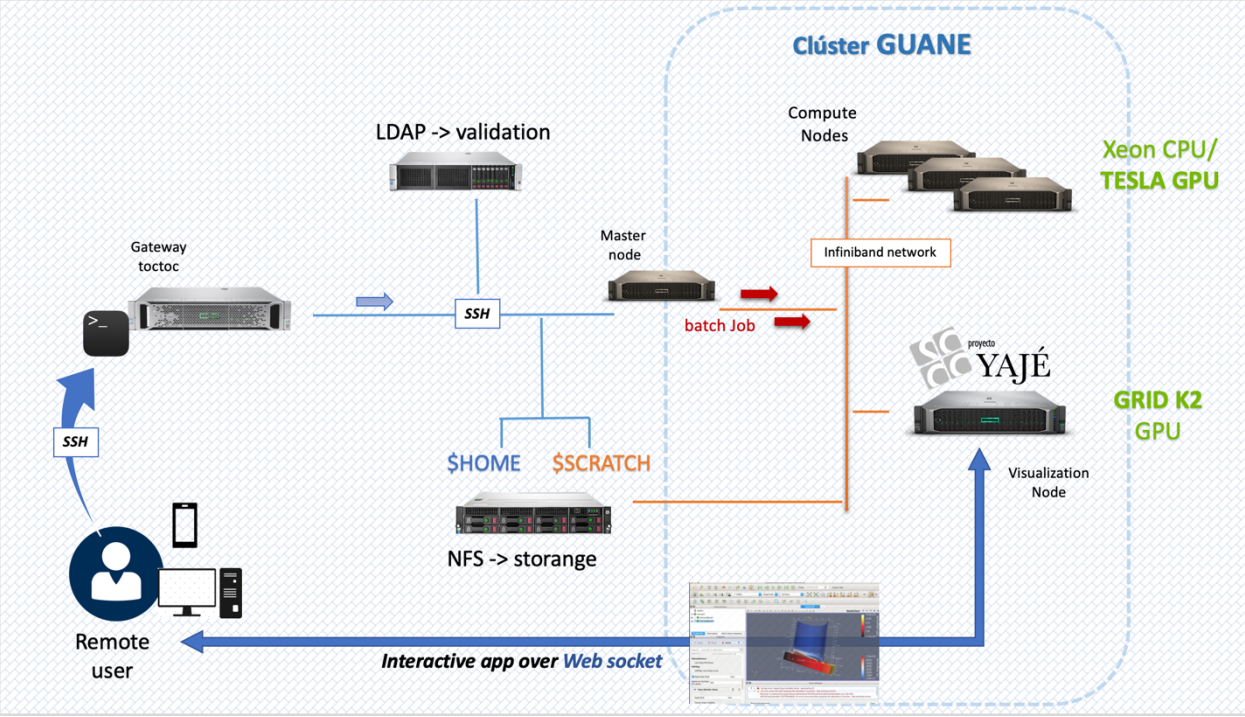

Figura 12 Topología de red de la infraestructura GUANE-1
Nota: Acorde a esta topología, los usuarios remotos logran tener acceso a los servicios de súper computo por sesión ssh y se les da la oportunidad de reservar servicios de computo con la red básica para trabajos livianos o la red infiniband para el manejo de big-data. La flecha doble azul representa la atracción de una conexión tipo web por webSockets y la plataforma YAJÉ-2.

6.1 Modelo de uso de la plataforma por usuario

Para facilidad del usuario se ha definido un diagrama de flujo que resume y permite identificar los pasos a seguir al momento de utilizar la plataforma. El aplicativo se diseñó tolerante a fallos de red, manteniendo un constante monitoreo sobre los servicios solicitados, y reiniciándolos en caso de desconexión o bloqueo de la aplicación. Figura 13

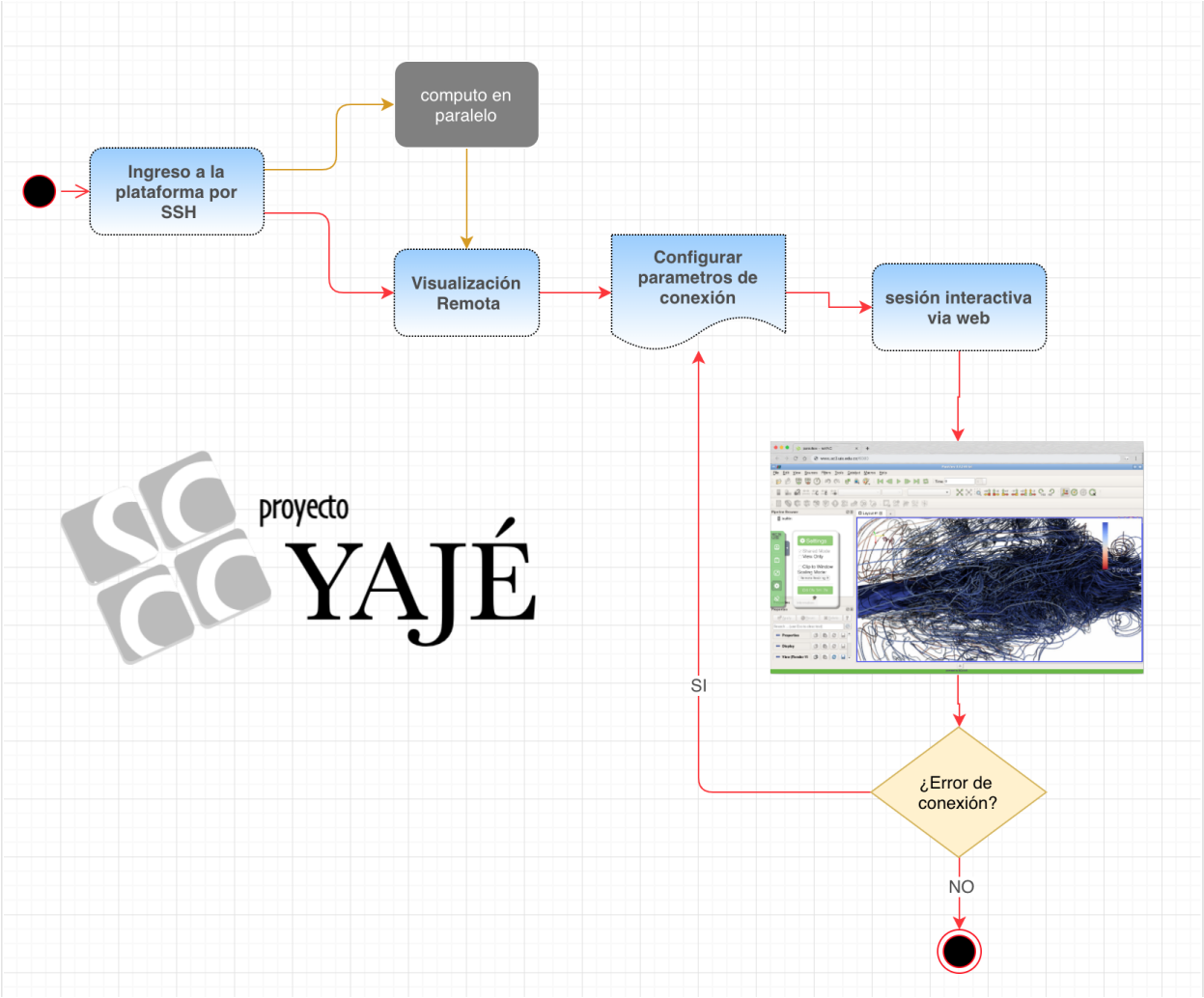

Figura 13 Diagrama de flujo uso de la plataforma, vista Usuario.

- El usuario accede por una sesión ssh, al clúster GUANE-1.
- En este punto, al usuario se le presenta la opción de acceder directamente a los recursos de visualización remota, o realizar un flujo de trabajo clásico de computación en paralelo sobre el clúster, y posteriormente, sin tener que mover los datos generados en previas simulaciones, solicitar una sesión de visualización.
- Se le presenta al usuario información relevante para el acceso web a la aplicación solicitada.

- La información entregada al usuario permite desde cualquier dispositivo móvil, retomar la sesión y/o compartirla a otros usuarios.

6.2 Modelo funcional de la plataforma de visualización.

Para ofrecer un servicio transparente y compatible a la arquitectura presente en el clúster GUANE-1, se implementó el acceso de la plataforma con los mismos principios de validación de usuarios y asignación de espacio de almacenamiento ya existentes.

La implementación del protocolo LDAP en el clúster permite asignar y mantener servicios de información por directorios de forma distribuida, accesible para todos los nodos, permite validar usuarios por credenciales SSH de forma transparente y trasversalmente a toda la infraestructura, lo cual ya se encontraba implementado en el clúster.

En la **Figura 14** a través de un diagrama de secuencia se define los mecanismos de comunicación y paso de mensajes entre los componentes del sistema desarrollado sobre una sesión de visualización establecida.

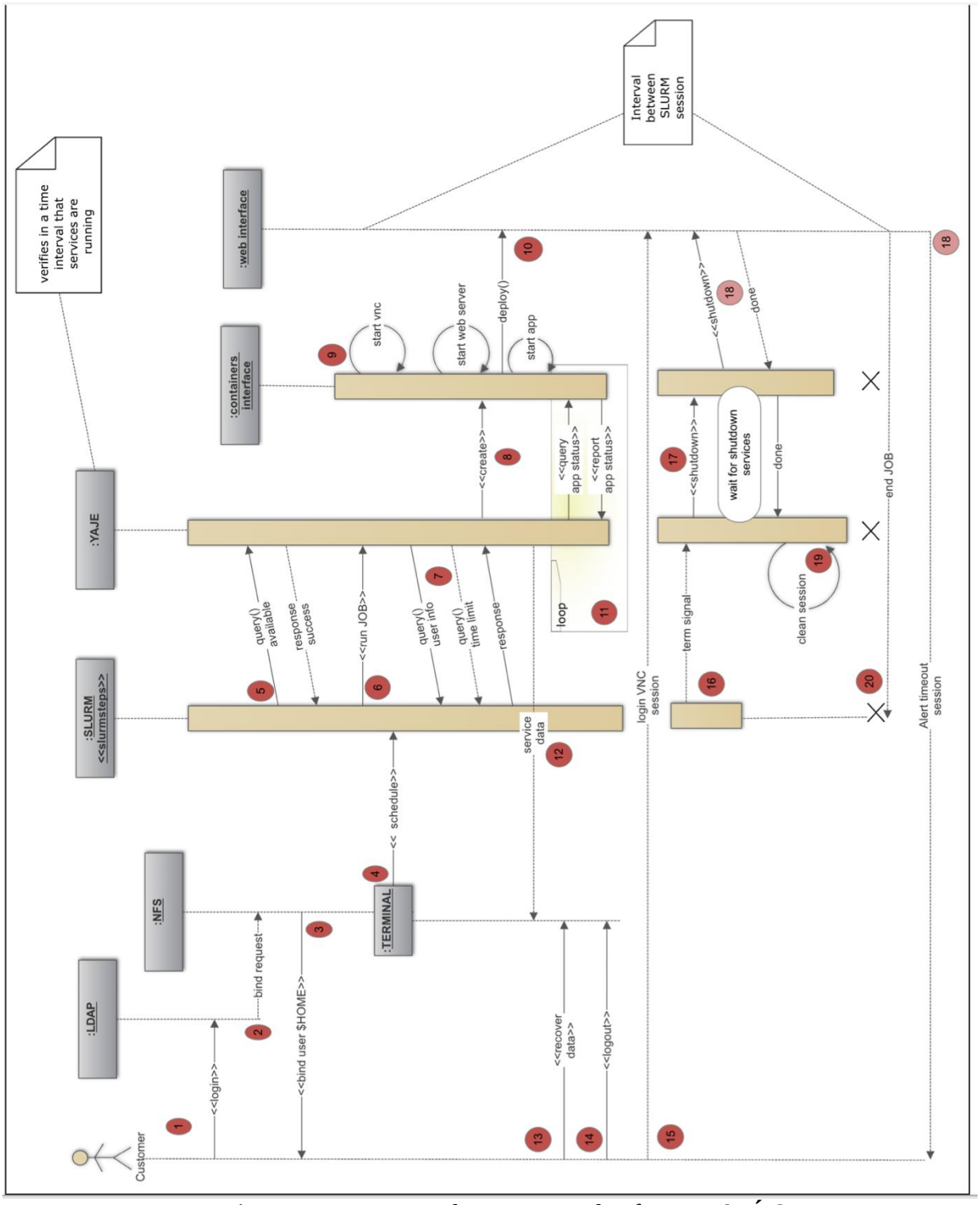

Figura 14 Diagrama de secuencia Plataforma **YAJÉ-2**

1. El usuario de forma remota a través de una terminal provee las credenciales de acceso a la infraestructura HPC por medio solicitud de acceso SSH.
2. El servidor LDAP valida al usuario con las credenciales ingresadas y se procede a través de consultas a un servidor NFS, a montar el directorio de usuario, para dar acceso al cliente sobre sus datos, programas y variables de entorno como servicio.
3. Se garantiza acceso a una terminal para que el usuario pueda solicitar servicios de computo distribuido o de visualización remota.
4. El usuario a través de un script Shell y el manejador de recursos SLURM, solicita recursos de visualización indicando la aplicación, la versión y metadatos apropiados para el manejador de recursos.
5. SLURM valida el script enviado por el usuario y procede a consultar recursos disponibles en el nodo de visualización remota YAJÉ.
6. Se envía el JOB solicitado al aplicativo Yajé.
7. La plataforma Yajé realiza una serie de consultas al manejador de recursos, buscando la información del usuario que requiere el servicio y el tiempo límite de sesión.
8. Se procede a crear 2 contenedores Singularity por usuario, el primero corresponde al entorno web y el segundo a la aplicación solicitada.
9. A través de la API de Python para Singularity spython (Sochat, 2018), como interfaz con el motor de contenedores, se procede a desplegar los micro-servicios dentro de cada contenedor, en un entorno aislado de los demás usuarios.
10. Se instancia un servidor web, un cliente vnc web, con los datos de la sesión SLURM, y se descubren los servicios solicitados a través de una URL con un puerto de conexión único por usuario.

11. Se realiza un control dado un determinado intervalo de tiempo, de los procesos desplegados, para garantizar su integridad y en caso de que existiere alguna falla o caída de servicio, se procede a levantar los procesos dentro de los contenedores de forma transparente.
12. Se le indica al usuario a través de un archivo de texto que corresponde al archivo de salida previamente definido en el script enviado al manejador de recursos, la URL en la cual puede acceder al servicio solicitado, junto con claves de acceso pertinentes y metodologías de enrutamiento en caso de requerir una conexión externa a la red UIS.
13. El usuario recolecta la información enviada por el nodo de visualización, en la cual se incluye un archivo de salida para errores, en caso de que exista alguno de ellos.
14. En este punto ya no es requerido mantener abierta la conexión por terminal al nodo maestro del clúster GUANE-1, por tanto, es viable terminar la sesión ssh, mientras los recursos solicitados siguen ejecutándose en el nodo de visualización como servicio.
15. El usuario en un cliente web (Chrome/Firefox/Safari) con capacidad html5 ingresa los datos proporcionados en el punto 13, para acceder a los recursos gráficos de alto rendimiento solicitados, a través de una interfaz web.
16. Después de que, transcurrido el tiempo de sesión solicitado, el manejador de recursos envía una señal de finalización de servicio al nodo YAJÉ.
17. Esta señal de finalización es propagada a todos los contenedores desplegados con un tiempo de gracia para finalizar correctamente los servicios.
18. Se da inicio a la fase de eliminación de procesos iniciando por los servicios web; al mismo tiempo se le avisa al usuario con un tiempo de 5 minutos para que salve su información.
19. Yajé, procede a eliminar archivos temporales, claves de usuarios, archivos de bloqueo de sesión y verifica que todos los servicios se han finalizado de forma adecuada.

20. SLURM como gestor y agente de los servicios solicitados, procede a finalizar con una señal SIGKILL, todos los procesos remanentes desplegados por el script del usuario, finalizando así la sesión de visualización remota.

6.3 Plataforma como un modelo de Ingeniería de Software.

En este apartado se presenta algunos modelos de ingeniería de software que, a modo conceptual, se han interrelacionado en la plataforma desarrollado, con esto se pretende sentar las bases para posteriores desarrollos, desde una perspectiva más ingenieril.

6.3.1 Contenedores como servicio. Para proveer un servicio flexible donde los desarrolladores e investigadores generen los contenidos necesarios para su propio trabajo, por medio de contenedores Linux , se implementó un concepto teórico de arquitecturas de diseño, adaptándolo a los elementos existentes en el clúster, así se propone un nuevo elemento como mezcla entre Aplicaciones como Servicio e Plataforma como Servicio, llamada Contenedores como Servicio CaaS, brindando una funcionalidad específica en búsqueda de una plataforma escalable, robusta y fácil de mantener. (Docker)

Adaptando estos elementos a la infraestructura subyacente en el laboratorio de Súper Computación, Clúster GUANE-1, se propone proveer un servidor de contenedores (nodo de visualización remota), liviano, adaptable que incluya todas las herramientas para desplegar contenedores. Figura 15

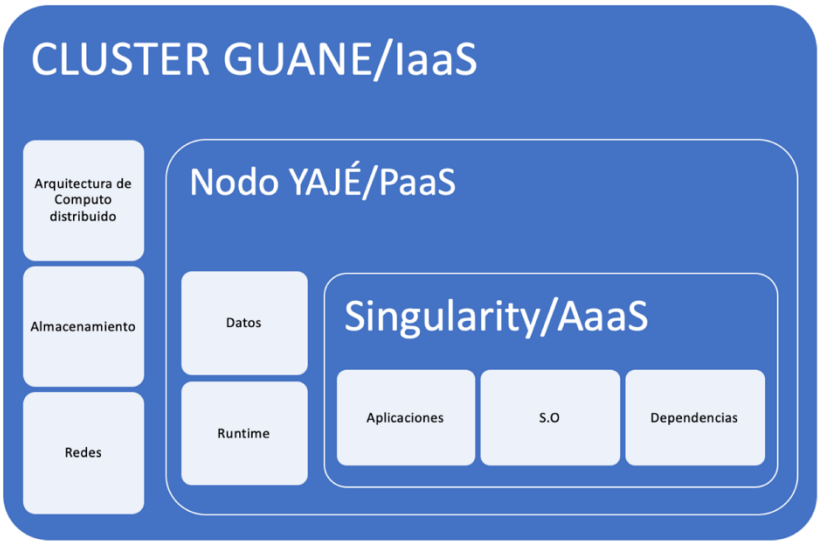

Figura 15 Modelo Integración de servicio GUANE-1 UIS.

6.3.2 Metodología DevOps Se propone implementar la metodóloga DevOps, para lograr una integración más transparente a la hora de usar la plataforma, o extenderla hacia nuevas funcionalidades por parte del personal de soporte técnico a través esta metodología ágil. (Lwakatare, November)

La selección de esta práctica de ingeniería del software de integración entre investigadores, desarrolladores y personal de soporte fue necesaria debido a que en la fase requisitos, al trabajar de forma interdisciplinar se detectó que los servicios ofrecidos en los entornos HPC, no se presentan de forma totalmente clara al usuario final; y este solo procede a seguir un algoritmo presentado por el personal técnico para la utilización de los recursos; no obstante los recursos no son totalmente aprovechados debido a la falta de experticia de los agentes humanos en el campo de trabajo de su contraparte.

Por lo anterior se propone DevOps para integrar al investigador en el desarrollo del software, según sus necesidades específicas; al arquitecto de soluciones en las tareas de soporte y al personal técnico facilitarle la automatización, el control de versiones y el monitoreo en todos los pasos del ciclo de vida de un software a través de la implementación completa de contenedores Linux en toda plataforma de súper computación existente, concretando así reingeniería de procesos para una arquitectura HPC. De esta forma queda abierta la posibilidad de que el usuario desarrolle sus propios contenedores Linux los defina en archivos Recipe-files, y enviarlas como especificaciones al grupo de desarrollo, los cuales buscaran automatizar los servicios y entregar retroalimentación tanto a usuarios como a personal de mantenimiento. Figura 16

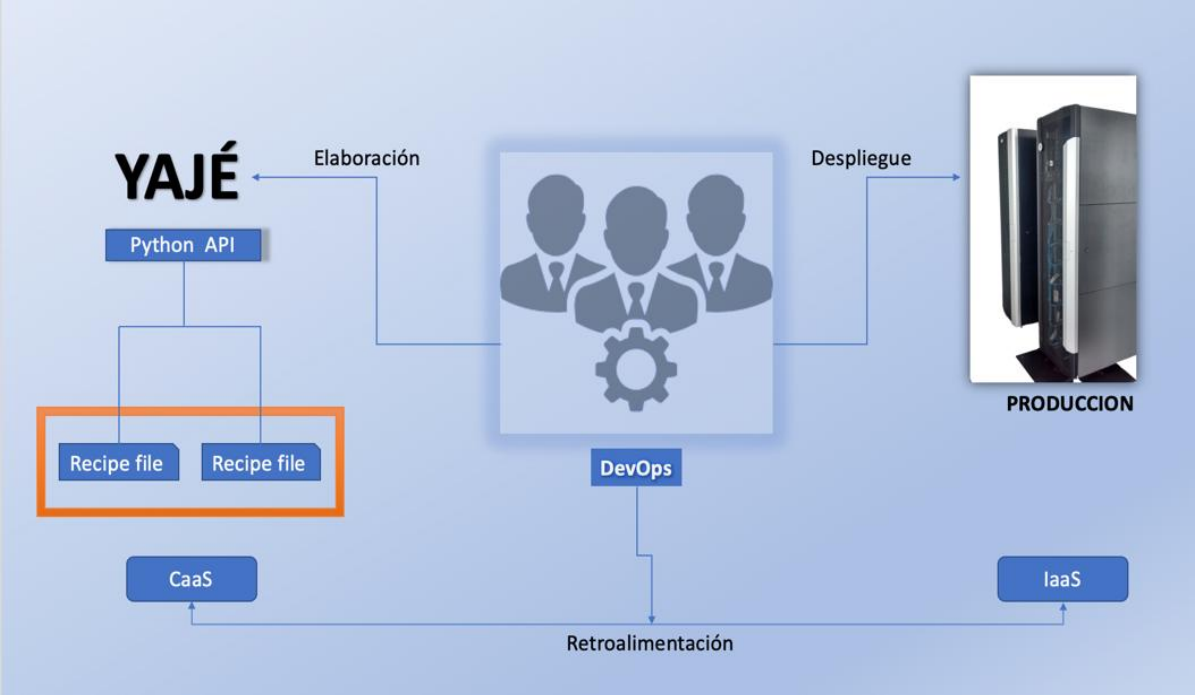

Figura 16 metodología DevOps a la infraestructura HPC GUANE-1 UIS

7. Resultados obtenidos y validación de la estrategia.

En este capítulo se da una breve introducción a los componentes desarrollados, los cuales se tratan en detalle de los componentes codificados, diagramación y procedimientos para trabajar con la herramienta desarrollada en la guía de usuario [apéndice A] y guía de programador [apéndice B].

7.1 Prototipo.

Para el desarrollo de la plataforma de visualización remota como un componente para la plataforma SC3, siguiendo los lineamientos estipulados de fácil integración, usabilidad y escalabilidad, se procedió a entregar un conjunto de paquetes de software orientados a la arquitectura HPC orquestado por un software que tiene como función automatizar el despliegue de servicios, el control de usuarios en un sistema multi-usuario y que presente una interfaz amigable de rápido uso.

Durante el transcurso de esta investigación se realizaron 3 prototipos funcionales, cada uno corresponde a la refactorización del anterior, y como agregado elementos de funcionalidad.

- Prototipo 1:
    - Desarrollado en batsh
    - Tecnología de contenedores Docker
    - Sin interfaz web.

Descartado por el lenguaje batch, no permitía implementar orientación a objetos, e implementar API de terceros.

- Prototipo 2:
    - Desarrollado en lenguaje Python 2.7
    - Tecnología de contenedores Docker
    - Con interfaz cliente vnc.

Este prototipo se descartó porque Docker tiene serios inconvenientes con la seguridad para entornos de producción, debido a que el usuario requiere obtener derechos de administrativos para desplegar contenedores.

- Prototipo 3:
    - Prototipo final
    - Desarrollado lenguaje Python 3.7
    - Interfaz web e integración con el clúster GUANE-1
    - Tecnología de contenedores Linux Singularity.

La interfaz web propuesta fue adaptada del proyecto noVNC, entregando únicamente las opciones mínimas de interacción al usuario en una sesión VNC, e integrando el servicio con scripts para aumentar la usabilidad del aplicativo. Figura 17

Figura 17 Prototipo 3, Interfaz Gráfica de Usuario

7.2 Codificación.

En la gestión de proyectos de software es imperativo que el equipo de trabajo posea herramientas base como plantilla para extender la funcionalidad de un proyecto de software, por ende, se presenta la estructura funcional para la creación de contenedores Singularity dentro de la plataforma YAJÉ, la codificación realizada de los entes de software desarrollados y diagramas para utilización del aplicativo.

La plataforma consta de tres módulos de software.

- Run_app, permite desplegar el contenedor de la aplicación solicitada.

Este módulo permite el despliegue del servidor VNC, el software solicitado, un script de inicio, y el software virtualGL para la gestión de llamadas openGL.

- Run_nvnc, permite desplegar el entorno web en un contenedor Singularity.

Integra el cliente web vnc, servidor web, modulo proxy websockify.

- Yajé (main), modulo principal de la aplicación controla e integra elementos funcionales como, puertos Unix, sockets de comunicación, control de procesos y tiempos de sesión.

El directorio NFS, o carpeta de usuario es de acceso transparente y para su implementación requirió la integración de la plataforma con el clúster GUANE-1, el software manejador de recursos y la validación de usuarios por medio del servidor LDAP.

La comunicación con el cliente web, se realiza por protocolo websocket sobre un canal de red sobre el protocolo TCP al contenedor web. Figura 18

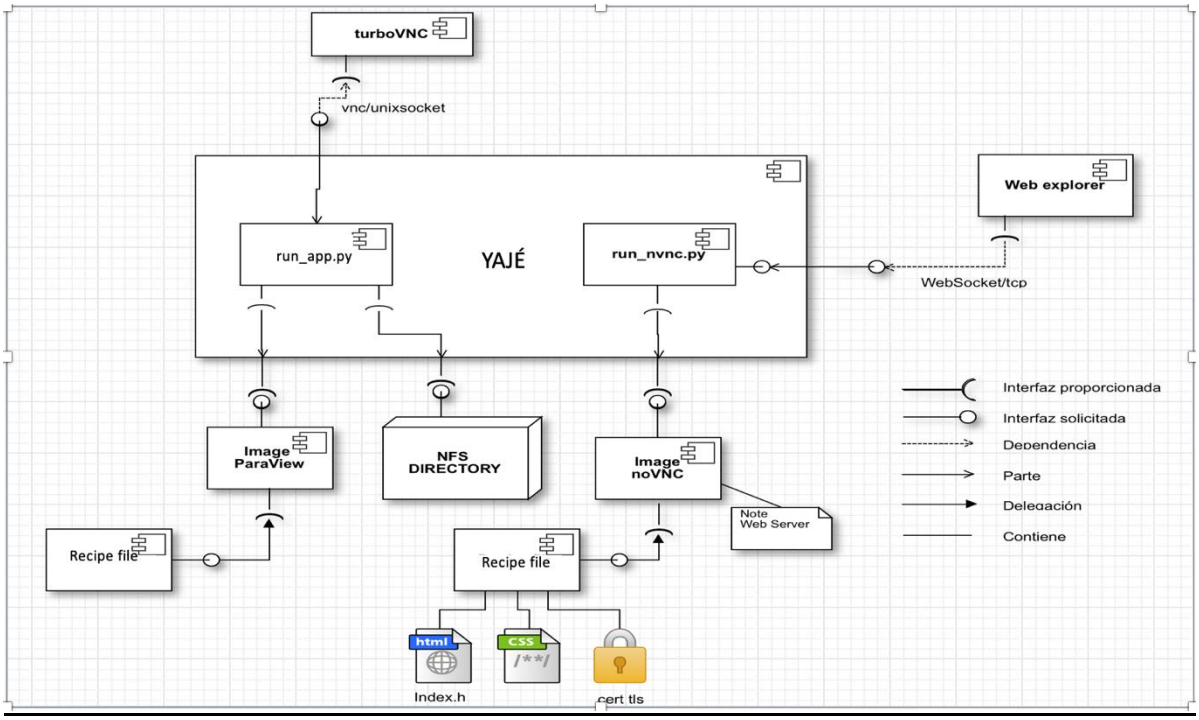

Figura 18 Módulos de software implementados para la herramienta desarrollada.

Con el fin de mantener la trazabilidad del desarrollo y facilitar la labor del equipo de soporte, la configuración inicial de la plataforma inicio con los pasos necesarios de instalación y puesta en marcha de software base del nodo de visualización, tales como, sistema operativo, drivers, dependencia de paquetes para la instalación de Singularity, y los demás softwares utilizados en esta investigación. [apéndice C]

A través de la implementación de control de versiones distribuido con GIT y GitLab se propone mantener la trazabilidad del código implementado y el desarrollo de toda la plataforma Figura 19

Figura 19 modelo propuesto de integración continua

7.3 Métricas de rendimiento.

Las métricas de rendimiento se dividieron en medición FPS en pruebas locales y pruebas remotas, pruebas de latencia, pruebas de estrés de hardware y prueba del caso real de simulación de dinámica de fluidos.

Se presentan los datos obtenidos con las métricas de rendimiento implementadas, y el análisis de cada prueba se discrimina en el capítulo 8 Discusión.

7.3.1 Cuadros por segundo FPS Como métrica inicial de rendimiento se tuvo en cuenta, la capacidad de renderizado del nodo de visualización sin interferencia de la red, medida como número de FPS, variando factores como el número de elementos gráficos muestreados en un determinado tiempo y las resoluciones más comunes en entornos de escritorio.

En esta fase se utilizó un benchmark llamado glxspheres64 (dcommander, 2017), el cual viene integrado en el paquete de software virtualGL y permite medir el rendimiento de aplicaciones que utilizan las librerías OpenGL para realizar tareas de renderizado por hardware. Figura 21

$ vglrun glxspheres64 -w 7680x4320 -n 100000
Parámetros -w = permite variar la resolución -n = varia el número de esferas (polígonos)

Figura 20 Comando para desplegar el benchmark

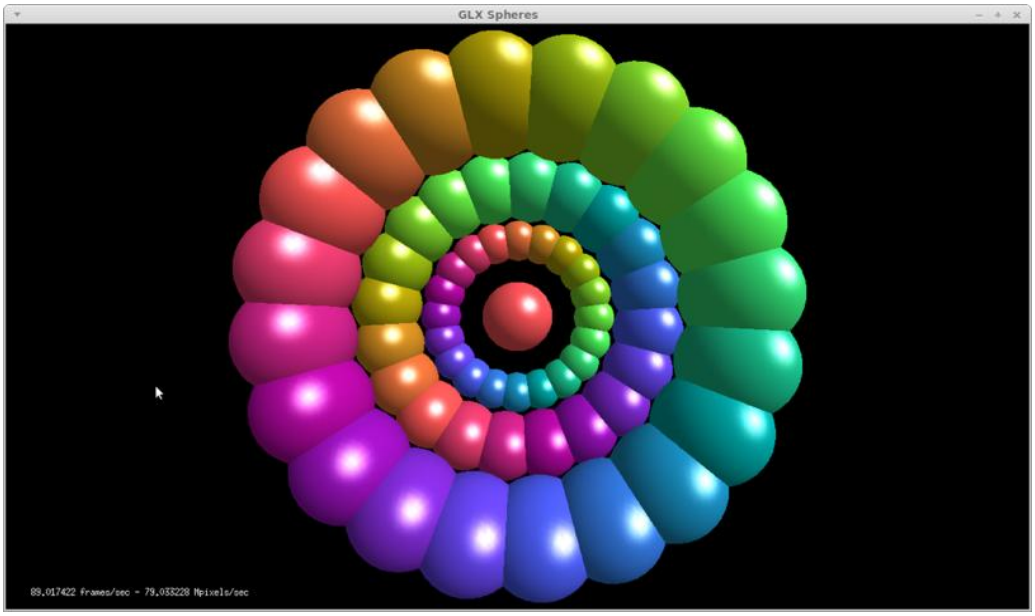

Figura 21 GUI benchmark glxspheres64

Como elemento de base comparación sin utilizar la tecnología de contenedores, se midió localmente la capacidad de renderizado en FPS con un software comercial de la empresa NICE llamado NICE Desktop Cloud Visualization, el cual usa el protocolo VNC en una versión modificada licenciada.

NICE Desktop Cloud Visualization es una tecnología de visualización remota que le permite a los usuarios de forma remota y segura conectarse a aplicaciones 3D hospedadas en servidores de alto rendimiento. (Amazon Web Services, Inc, 2018)

Este software fue obtenido como paquete binario **nice-dcv-** en la última versión disponible **2017.1 r5870**. Nota: número de polígonos representados como número de esferas renderizadas, tomando de base 1024 polígonos por esfera. Tabla 8

Tabla 8 *Comparativa cuadros por segundo [FPS] a distintas resoluciones.*

		# esferas					
resolucion	NICE	61	100	200	1,000	10,000	100,000
720p	585.483	565.320	502.536	345.823	162.749	25.823	2.793
1080p	566.573	296.425	256.280	249.831	128.598	25.763	2.632
1440p	564.486	157.241	162.750	93.255	92.702	22.970	2.699
2160p	572.821	84.894	83.761	75.021	57.921	14.890	1.740
4320p	572.085	21.652	22.012	22.015	17.941	5.267	0.563

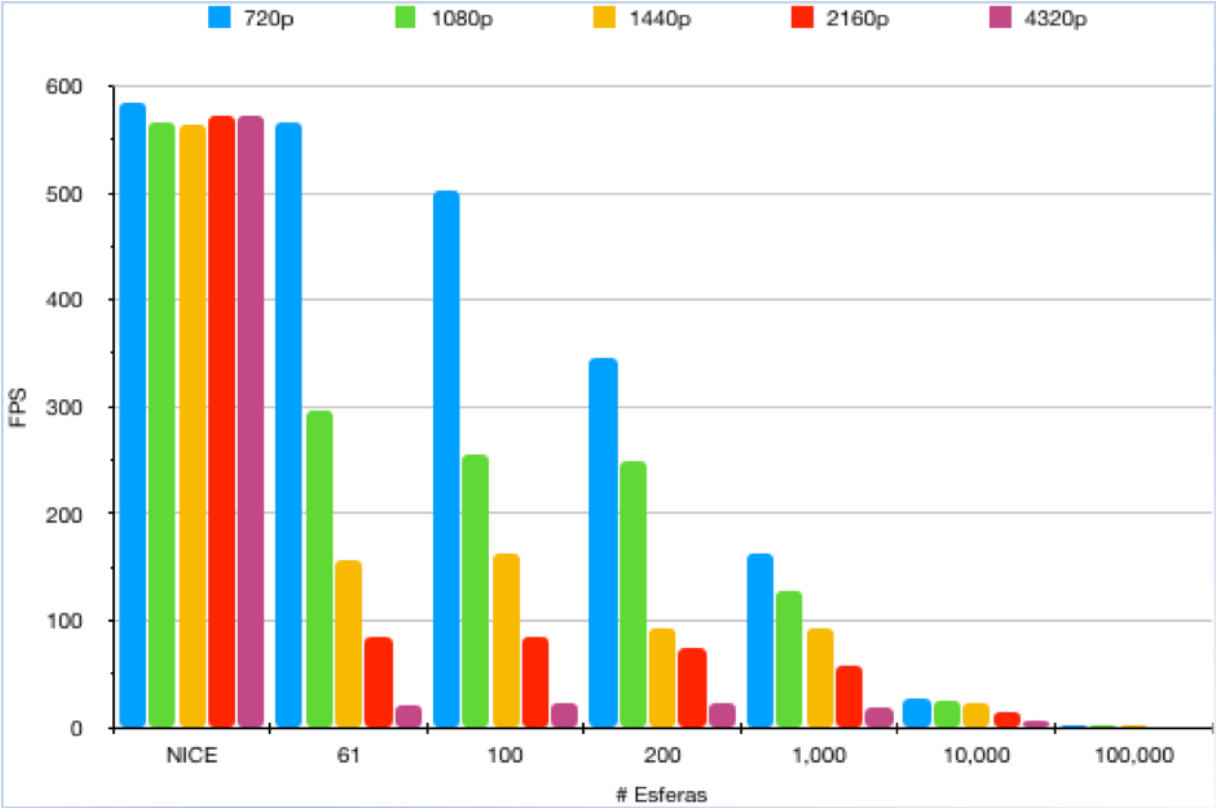

Figura 22 Número de FPS contra número de esferas renderizadas a distintas resoluciones.

Análisis:

Se evidencia que el benchmark NICE genera aproximadamente la misma calidad de FPS para todas las resoluciones de pantalla, esto se debe a que este solo permite variar la resolución de pantalla y no el número de polígonos representados en la escena, por tanto, no aumenta la complejidad de las imágenes renderizadas.

El benchmark comercial implementa la tecnología de compresión de imágenes por hardware lo cual le permite mantener el rendimiento medido por FPS más alto que otras implementaciones, mientras la aplicación desarrollada implementa compresión de imágenes por software.

En esta prueba se concluye que la tecnología de contenedores Linux no se presenta un elemento que limite el rendimiento de una aceleración por hardware, puesto que no agrega capas de software emulado a la implementación.

La implementación del protocolo VNC tiene un rendimiento óptimo en resoluciones de pantalla que sean inferiores a 2160p.

Variando la resolución de escalado de pantalla y la complejidad de datos 3D requeridos al momento de realizar la medición representados, se observa una tendencia decreciente en la cantidad de FPS entregados, lo cual es un comportamiento esperado al existir mayor cantidad de información gráfica que necesita ser renderizada; este mismo comportamiento se presenta al mantener constante el número de polígonos y aumentar gradualmente la resolución de muestreado.

7.3.2 Tiempo de respuesta en una sesión interactiva.

Diferentes estudios únicamente enfatizan en la medida de FPS como métrica de rendimiento, sin embargo, esta métrica no es suficiente para escenarios donde el usuario interactúe con una interfaz gráfica dada la latencia entre el tiempo de renderizado y el tiempo en el que el software cliente recibe esa información para ser procesada en la pantalla; en este contexto una medida de rendimiento en entornos interactivos más apropiada es el así denominado tiempo de respuesta a eventos de usuario.

El factor de medición en este contexto se define como latencia de red; se probaron las dos configuraciones de red, local y acceso remoto a la plataforma, sobre un grupo de datos de prueba para el software Paraview.

La herramienta de medición corresponde es un software integrado al buscador de internet Google Chrome, el cual permite perfilar el rendimiento de carga, FPS, latencias de una página web, consumos porcentuales de CPU, transferencias de red, entre otros.

Caso: Un flujo de glóbulos blancos a través de un segmento de arteria, y se midió los FPS entregados por la aplicación en una interfaz web desarrollada.

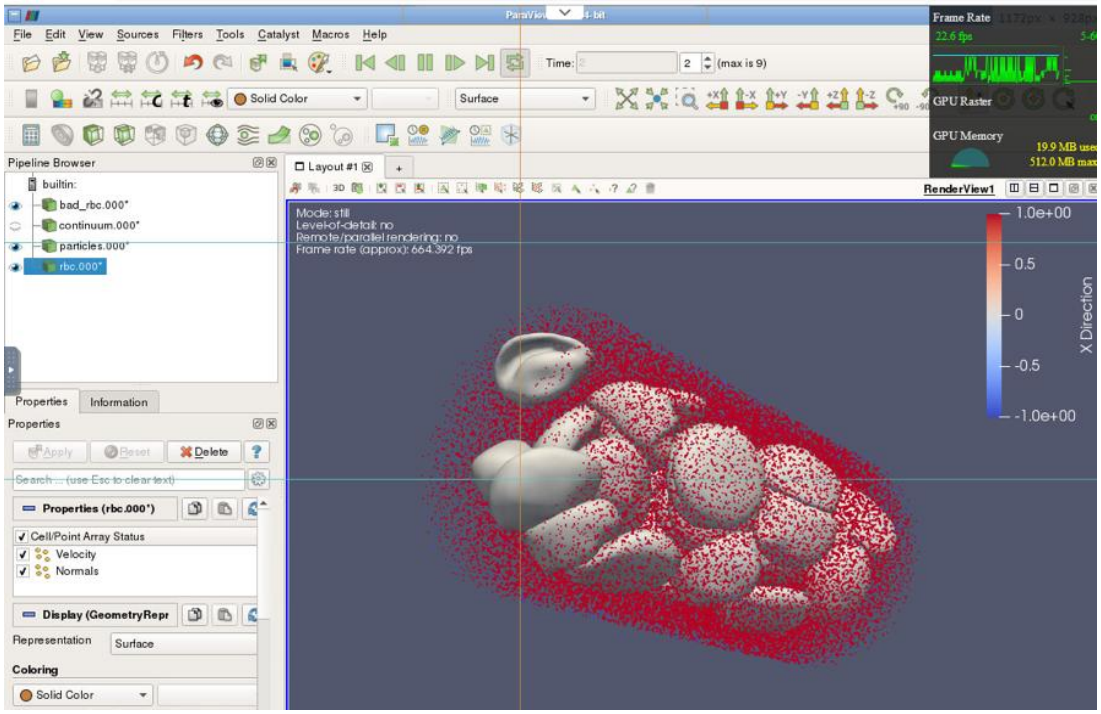

Figura 23 simulación flujo de glóbulos en un segmento de arteria

Tabla 9 *Comparativa prueba de rendimiento, factor latencia de la red*

Caracterisitica	Red local	Red externa
Ancho de banda [Mbps]	200	10
Latencia media (ICMP PING) [ms]	3.2	55
[Fps] medidos en el cliente	22.6	4.7
[Fps] medido en el servidor	664.392	615.362
Tiempo de medición [minuntos]	15	15
Información transferida [MB]	40	35
Consumo de cpu cliente [%]	30	12

Análisis:

La capacidad de rendimiento analizada según los FPS, medida para conexión local y conexión remota, evidencia que el factor preponderante al momento de ejecutar una simulación corresponde a la latencia de la red, puesto que, aunque se trabaja con una diferencia de 190 Mbps como ancho de banda entre las dos pruebas, el frame-rate medido desde la aplicación cliente (el buscador de internet) se duplicó en la conexión local de baja latencia, entregando así una simulación más fluida.

Para las dos pruebas realizadas se evidencia un bajo consumo de ancho de banda, debido a que la tecnología VNC únicamente transmite por red las porciones de imagen que cambian en una simulación, mientras otros segmentos de la imagen permanecen invariantes y no son reenviados al cliente.

Los valores obtenidos en esta prueba por FPS dependen de la variación de la imagen representada, lo cual ocurre en dos casos, la interacción del usuario con la sesión remota a través de eventos de mouse o de teclado y el caso en que se esté representado una animación; esto debido a que VNC se diseñó con el fin de adaptarse a redes tanto de baja y alta latencia; por tanto, solo se puede detectar variación de FPS al momento de enviar y/o. recibir eventos al servidor.

El consumo de ciclos CPU del lado del cliente, medido en redes de baja latencia es más elevado que en redes de alta latencia, debido a que el servidor VNC se adapta enviar imágenes comprimidas a una frecuencia más alta, tomando ventaja de la baja latencia de la red, así hay un flujo imágenes mayor que necesitan ser descomprimidas antes de ser procesadas por el cliente.

8. Discusión.

En esta sección se presenta el análisis de la herramienta dando una discusión sobre la implementación e integración de YAJÉ en el clúster GUANE-1.

8.1 Métricas

Las métricas de rendimiento fueron tomadas como prueba de concepto, pero debe tenerse en cuenta que el rendimiento que la solución depende de muchos factores, como tipo de CPU, tipo de red implementada, lo que implica que los benchmark realizados en este aspecto no son

definitivos y es fuertemente recomendable correr benchmarks para cada aplicación virtualizada cuando entre en consideración la puesta en marcha para entornos HPC.

8.2 Implementación de la plataforma

La implementación de contenedores Linux, aparte de simplificar la infraestructura y compartir de forma más eficiente recursos de computo, permite que coexista versiones de software que son ortogonales entre ellas, pues el aislamiento de recursos pilar de la filosofía de contenedores, gestiona estos aspectos de manera muy eficiente.

En el transcurso del proyecto de logro aportar al desarrollo de la API Python de Singuarity, a través de código propio, seguimiento de errores, y soporte a usuarios.

Para asegurar el acceso a la plataforma y más exactamente a los contenedores con interfaz gráfica, se implementó autenticación de usuario para el acceso a la sesión VNC; también se implementa túneles SSH y cifrado TLS para el no repudio de la información y verificación de identidad del lado del servidor.

Se ofrece un servicio web, con la implementación del cliente VNC web noVNC, que permite a través de la mayoría de los buscadores modernos como Mozilla Firefox, Google Chrome, Apple Safari con capacidades HTML5, ofrecer elementos suficientes para el acceso del cliente a entornos HPC. Este aspecto depende en gran medida del grado de soporte las primitivas HTML5.

La solución también es accesible a través de dispositivos móviles como Tablet, Smartphone, etc. Esto debido a que HTML5 de igual manera es soportado por versiones móviles de los buscadores y noVNC está adaptado para ofrecer servicio móvil.

Como elemento fehaciente en el uso de contenedores por medio de Singularity, se facilita al equipo de desarrollo, generar nuevos contenidos y desplegar aplicaciones adicionales con mínimos cambios en la arquitectura. Esto se concretó al implementar la API Python de Singularity, documentación de código y generación de listas de comandos completos desde la instalación del sistema operativo hasta la puesta en marcha de la plataforma; así mismo se trabajó con una perspectiva DevOps que integre al equipo de desarrollo, soporte de usuarios e investigadores.

La plataforma se desarrolló sin límite de usuarios, pero está limitado a los recursos de hardware disponibles, ancho de banda por consumido por usuario, nivel de compresión de datos y posteriores refinamientos de la plataforma.

Durante el transcurso de la investigación se pudo conocer como el sector informático y las empresas vanguardistas prestan el servicio de visualización remota, y se buscó la correlación de tecnólogas de punta para ofertar un servicio de alto rendimiento, fácil de usar, mantener y escalar; el servicio más utilizado en entornos de visualización remota para HPC, lo presta la empresa NICE, con su implementación web de visualización y protocolos privativos.

Se implementa una plataforma libre de limitaciones por conceptos de licencia de software.

La plataforma desarrollada mediante este trabajo de investigación ya se encuentra en fase de pruebas públicas, y se presenta con tres aplicaciones como servicio, Paraview, Blender y VisiIT.

9. Conclusiones

Durante el transcurso de esta investigación se logró concretar los siguientes elementos:

- La plataforma desarrollada como servicio de visualización remota **YAJÉ-2**, permite a usuarios con recursos de hardware limitados tener acceso a aplicaciones que requieren un entorno gráfico de alto rendimiento para ser ejecutadas; de esta forma logro aporta a los investigadores que utilicen la visualización remota como parte de su flujo de trabajo.
- Se concreto el diseño y codificación de una plataforma de visualización remota, la cual se encuentra operativa dentro de la infraestructura GUANE-1.

Con la creación y puesta en marcha de la plataforma de visualización YAJÉ-2, se da por cumplido el objetivo específico número 1, creación de una plataforma prototipo de visualización remota.

- Con la implementación de la visualización de un caso real como prueba piloto de la plataforma, se redujo el tiempo necesario para la etapa de post procesamiento en una simulación, demostrando así las ventajas en cuanto a rendimiento de la infraestructura remota subyacente.

Se logro implementar y visualizar un caso real de simulación de dinámica de fluidos, evidenciando las ventajas ofrecidas por la plataforma remota y con esto se da por cumplido el objetivo específico número 2, implementación de la visualización de una simulación de flujo turbulento.

- Se logro integrar la propuesta desarrollada de forma transparente a los recursos de súper computo ofertados por el centro de súper computo SC3-UIS.

Se diseño e implemento modelos de acoplamiento escalable a la infraestructura HPC existente, permitiendo así generar una nueva línea de desarrollo robusta, fácil de mantener y escalar; con esto se da por cumplido el objetivo específico número 3, sobre la creación de una estrategia de acoplamiento de la aplicación YAJÉ con la infraestructura clúster GUANE UIS.

Se logro generar métricas de rendimiento, teniendo en cuenta las limitaciones de la plataforma desarrollada y se comparó con plataformas existentes, evidenciando la creación de un producto viable para entornos de producción, el cual ya se encuentra operativo y en uso por diferentes investigadores dentro de la Institución; con esto se da por cumplido el objetivo específico número 4, comparar los resultados obtenidos y validar la estrategia seleccionada.

Se logro entregar una plataforma con tecnologías de punta en el ámbito opensource, a la par de centros de investigación internacionales.

- Las pruebas de rendimiento e interacción con la plataforma permitieron evidenciar que se requiere un ancho de banda recomendado de 10 [Mbps], y una latencia de red inferior a 50 [ms] para acceder a los recursos de computo de forma remota, esto para garantizar buena experiencia de usuario.

10. Recomendaciones y trabajo futuro.

- Implementar un sistema de pantallas acopladas de visualización, debido a que renderizaciones de alta calidad requieren de un esquema de alta resolución para ser interpretadas correctamente sin pedida información relevante.

- Implementar un mecanismo para adaptar el número de FPS entregadas por el servidor VNC, debido a que entre más información sea enviada hacia el cliente, más ciclos de procesador son necesarios para decodificar esta información recibida.

- Implementar un módulo que permita del lado del cliente, adaptar la taza de refresco y calidad de imagen recibida, así mismo visualizar en tiempo real, ancho de banda consumido, latencia de red, y FPS.

- Extender la plataforma de visualización a otras aplicaciones enfocadas al uso de GPGPU y ofertar una mayor cantidad de productos que aprovechen la totalidad de los recursos disponibles.

- Promover la extensión del nodo de visualización hacia un Clúster dedicado para visualización, dentro del laboratorio de supercomputación de la universidad.

- La implementación de una arquitectura de almacenamiento distribuido puede acelerar el renderizado de aplicaciones al disminuir los tiempos de acceso a disco y minimizar los cuellos de botella que presenta un modelo de almacenamiento centralizado; se recomienda el uso de Lustre como sistema de archivos.

- Se recomienda seguir los lineamientos estipulados en la guía de programador para extender las funcionalidades de la herramienta desarrollada.

- Desarrollar y poner en marcha una plataforma web que integre todos los servicios ofertados por el laboratorio de súper computación, y permita a través de esta realizar tareas como, desplegar trabajos, reportes de resultados de las investigaciones, gestión de usuarios, solicitar recursos de visualización, etc.

- Integrar al equipo de desarrollo con el equipo de soporte para lograr mayor fluidez en la comunicación de estos, y agilizar la entrega de productos de calidad, esto es la ideología del modelo DevOps de ingeniería del software, que también puede ser aplicado a entornos HPC.

- Se recomienda la utilización de la plataforma de visualización remota para redes de baja latencia inferior a 50 [ms] y un ancho de banda superior a 10 [Mbps].

11. Limitaciones.

- Las especificaciones de hardware disponibles como RAM, disco duro, son un factor preponderante para tareas de renderizado, puesto que el consumo de memora volátil escala exponencialmente llegando a requerir en poco tiempo cientos de Gigabytes de memoria; y Terabytes de almacenamiento, las latencias más frecuentes se vieron en el acceso a disco de almacenamiento para la lectura y escritura de datos, con alta dependencia del sistema de archivos, el software de lecto-escritura seleccionado, en el caso de prueba PallelReader, un software que implementa ParaView al realizar lecturas en paralelo.

- Las aplicaciones utilizadas para el caso de prueba se enfocan en explotar el paralelismo de tareas a nivel de memoria de procesador y no hacen uso extensivo de los núcleos de dispositivos de procesamiento grafico dedicado GPUs; esto ocurre porque la única forma viable de paralelizar el modelo CFD en este software, sin reescribir los solvers componente, enfocándolos a la implementación de CUDA, es por medio de rutinas MPI/OPENMP. Existen plugins que optan por unificar recursos de memoria RAM y GPU, pero son limitados en sus versiones gratuitas y requieren modificar el caso de estudio; para casos particulares, algunos solvers no han sido paralelizados para entornos GPGPU.

- Implementar contenedores es una metodología solo apta para aplicaciones que puedan desplegarse sobre el núcleo Linux, aplicaciones que corren sobre otros Sistemas

Operativos no son viables en este concepto, requiriendo otra perspectiva de implementación como máquinas virtuales.

- Ancho de banda y carga útil de red.

Aunque el objetivo es prestar un servicio con la menor carga de información enviada por red, siempre está presente como limitante el ancho de banda de red y la latencia que posea el usuario que accede a los recursos, y la carga de trabajo en el clúster de computo.

Referencias Bibliográficas

Amazon Web Services, Inc. (2018). *NICE Desktop Cloud Visualization: User Guide*. Recuperado el September de 2018, de aws: https://docs.aws.amazon.com/dcv/latest/userguide/dcv-ug.pdf

amonakov. (August de 2015). *primus*. Recuperado el November de 2018, de github: https://github.com/amonakov/primus

Bhaskaran, R., & Collins, L. (s.f.). *Introduction to CFD Basics*. Obtenido de cornell: http://dragonfly.tam.cornell.edu/teaching/mae5230-cfd-intro-notes.pdf

Center, A. C. (1 de January de 2009). Recuperado el September de 2018, de UIC: http://www.uic.edu/depts/accc/software/exceed/sshexceed.html

Cygwin Corporation. (2018). Recuperado el November de 2018, de cygwin: http://www.cygwin.com/

dcommander. (2017). *virtualGL*. (virtualGL, Productor) Recuperado el November de 2018, de github: https://github.com/VirtualGL/virtualgl/blob/master/glxdemos/glxspheres.c

Docker. (s.f.). *Delivering Containers-as-a-Service (CaaS) With Docker Datacenter*. Recuperado el November de 2018, de docker: https://www.docker.com/sites/default/files/Docker%20Datacenter_Datasheet.pdf

Ermakov, A., & Vasyukov, A. (April de 2017). *Testing Docker Perfomance for HPC Applications.* Paper, Mipt. Recuperado el november de 2018, de arxiv: https://arxiv.org/pdf/1704.05592.pdf

Ermakov, A., & Vasyukov, A. (2017). *Testing Docker Performance for HPC Applications.* Paper, Mipt.

ESI Group. (2018). *Open CFD toolBox*. Recuperado el October de 2018, de openfoam: https://www.openfoam.com/

FIRESMITH, D. (25 de September de 2017). Recuperado el APRIL de 2018, de insights: https://insights.sei.cmu.edu/sei_blog/2017/09/virtualization-via-containers.html

Forero, L. C., Uribe, R., Orostegui, S., & Mantilla, M. (January de 2011). *A lightweight and Dedicated Architectures Integration on a Platform for High Perfomance.* Obtenido de researchgate: https://www.researchgate.net/publication/215607999_GridUIS-2_A_Lightweight_and_Dedicated_Architectures_Integration_on_a_Platform_for_High_Performance_and_Scientific_Computing

Kitware. (2018). Recuperado el November de 2018, de paraview: https://www.paraview.org/

Kvm Project. (2018). *Linux KVM*. Recuperado el August de 2018, de https://www.linux-kvm.org/page/Main_Page

Lwakatare, L. E. (November). *DEVOPS ADOPTION AND IMPLEMENTATION IN SOFTWARE DEVELOPMENT PRACTICE: CONCEPT, PRACTICES, BENEFITS AND CHALLENGES.* (J. PRINT, Ed.) Oulu, Finland: University Oulu.

Martin, J. (s.f.). *novnc*. Recuperado el August de 2018, de https://novnc.com/info.html

Mott, R. L. (1996). *Mecánica de fluidos aplicada* (Vol. 4). (P. Educación, Ed., & C. R. Carlos, Trad.) Mexico, Mexico. Obtenido de wordpress: https://henryloaisiga.files.wordpress.com/2014/01/mecc3a1nica-de-fluidos-aplicada-mott.pdf

Nvidia. (June de 2014). *Nvidia.* Recuperado el 2018, de https://www.nvidia.com/content/pdf/remote-viz-tesla-gpus.pdf

Packard, K. (s.f.). *Getting X Off The Hardware.* Recuperado el November de 2018, de https://keithp.com/~keithp/talks/xserver_ols2004/xserver_ols2004.pdf

Parada, C. D. (2017). Arquitectura de visualización para aplicaciones cientificas como servicio. Bucaramanga, Colombia.

Riaño, N. D. (2017). Computación de Alto Rendimiento para la Visualización e Interacción Remota de Aplicaciones Científicas. Bucaramanga, Santander, Colombia.

Richardson, T., Stafford-Fraser, Q., Wood, K. R., & Hopper, A. (1998). *Virtual Networking Computing.* The Olivetti & Oracle Research Laboratory. IEEE Internet Computing.

Rojas, A. J., Vasquez, A. B., & Suarez, J. (2014). *wiki*. Obtenido de sc3: http://wiki.sc3.uis.edu.co/images/a/a2/G13.pdf

SC3 UIS. (20 de January de 2017). *Cluster GUANE*. (U. CAGE, Productor) Recuperado el November de 2018, de sc3: http://wiki.sc3.uis.edu.co/index.php/Cluster_Guane

Sochat, V. (2018). *Spython API*. Recuperado el November de 2018, de github: https://github.com/singularityhub/singularity-cli

The Apache Software Foundation. (2018). Recuperado el November de 2018, de Apache Guacamole: https://guacamole.apache.org/

Ullah, S., & Xuefeng, Z. (s.f.). Cloud Computing: A Prologue. Beijin, China.

VirtualGL Project. (11 de January de 2018). Recuperado el September de 2018 , de The VirtualGL Project: https://www.virtualgl.org/

Vmware. (2006). Recuperado el September de 2018, de vmware: https://www.vmware.com/pdf/virtualization.pdf

Webmaster. (25 de september de 2018). Recuperado el November de 2018, de Khronos: https://www.khronos.org/opengl/wiki/FAQ#What_is_OpenGL.3F