DISEÑO DE UNA ARQUITECTURA ESCALABLE PARA BIG DATA SOPORTADA POR SC3

JOSYMAR GARCIA ACEVEDO
JUAN EDUARDO ARIAS VALERO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2014

DISEÑO DE UNA ARQUITECTURA ESCALABLE PARA BIG DATA SOPORTADA POR SC3

JOSYMAR GARCIA ACEVEDO
JUAN EDUARDO ARIAS VALERO

TRABAJO DE GRADO PARA OPTAR AL TÍTULO DE INGENIEROS DE SISTEMAS E INFORMÁTICA

Director
Ph.D. RAÚL RAMOS POLLÁN
DOCTOR EN INGENIERÍA INFORMÁTICA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2014

DEDICATORIA

A Dios por ser siempre ese sentimiento de alegría, tranquilidad y serenidad en cada momento de esta etapa próxima a culminar, espero ser digno por tan valioso esfuerzo.

A mi padre Albeiro Garcia, a mi madre Consuelo Acevedo, no hay un día en el que no le agradezca a Dios el haberme colocado entre ustedes, la fortuna mas grande es tenerlos conmigo y el tesoro mas valioso son todos y cada unos de los valores que me han inculcado.

A mi hermano Ederson Garcia, gracias por servir de guía, por acompañarme siempre y más te agradezco por ser mi amigo.

A mis compañeros de universidad, por los gratos momentos, grandes recuerdos los que hoy quedan, los mejores deseos hoy y siempre.

A esa persona especial, por agregarle tranquilidad y alegría a mi vida, A.D.

A mis compañeros de selección, un honor haber tenido la oportunidad de compartir grandes momentos junto a ustedes, todos vestidos bajo una misma camiseta, camiseta que siempre llevare con orgullo.

Al profesor Raul Ramos por habernos ofrecido esta gran oportunidad que nos presenta un futuro promisorio.

Gracias.

Josymar Garcia Acevedo

DEDICATORIA

En primer lugar a Dios y a la vida, por permitirme cumplir uno de de tantos sueños.

A mi padre Juan Bautista, a mi madre Maria del Carmen, a mi hermana Carmen Ligia, humildes guerreros a los cuales les agradezco infinitamente todo su sacrificio para sacar adelante a nuestra familia, por quienes hoy en día soy la persona que soy, personas a las que amo con toda mi alma.

A mis amigos de universidad por acompañarme en esta aventura que tanto disfrutamos y sufrimos, Cindy Yulitze, Henry Abril, a mis amigos de ingeniería de sistemas y muy especialmente a Laura Vásquez la mujer que me acompaña en este momento tan especial de mi vida.

Al profesor Raul Ramos por habernos ofrecido esta gran oportunidad que nos presenta un futuro promisorio.

Gracias totales.

Juan Eduardo Arias Valero

Índice general

Índice de figuras

Índice de cuadros

Índice de Anexos

RESUMEN

Titulo: DISEÑO DE UNA ARQUITECTURA ESCALABLE PARA BIG DATA SOPORTADA POR SC3[1]

Autores: GARCIA ACEVEDO Josymar, ARIAS VALERO Juan Eduardo[2].

Palabras Clave: BigData, extracción de conocimiento, infraestructura de supercomputo, SC3, procesamiento a gran escala, grandes volúmenes de datos.

Descripción: Gracias a los grandes avances en cuanto a tecnologías de la información se refiere, las organizaciones han tenido que afrontar un gran desafió al momento de analizar, descubrir y entender la gran proliferación de información proveniente de todos los procesos realizados interna y externamente. Esta información no puede ser procesada óptimamente por sus herramientas tradicionales de computo y además pueden llegar a ser de cientos de petabytes. De manera que las nuevas herramientas adaptadas al procesamiento de estos grandes volúmenes de datos requieren de una gran velocidad de respuesta, paralelo a una rápida obtención de resultados que sean correctos y precisos. Estas son las características principales de oportunidad para Big Data.

La Universidad Industrial de Santander dispone de una infraestructura de computo administrada en parte por el grupo de investigación SC3, la cual se encuentra integrada a través de una plataforma denominada “GridUIS-2”. “GridUIS-2” integra y conecta diferentes recursos que se encuentran distribuidos en el campus central y la sede Guatiguará. Entre estos recursos tecnológicos se encuentra un supercomputador (guane1), dando así un servicio de computación de alto rendimiento a la comunidad universitaria y científica.

Teniendo en cuenta la gran disposición de estos recursos tecnológicos en la Universidad Industrial de Santander, nace la iniciativa al querer integrar y adaptar estas nuevas herramientas de computo al SC3, y así disponer de un pilar importante en cuanto a tecnologías de la información y supercomputación se refiere, como lo es Big Data, para el tratamiento de grandes volúmenes de datos.

[1]Trabajo de Grado con modalidad de Trabajo de Investigación.

[2]Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: PhD Raúl Ramos Pollán

ABSTRACT

Title: DESIGN OF A SCALABLE ARCHITECTURE FOR BIG DATA SUPPORTED BY THE SC3[3]

Authors: GARCIA ACEVEDO Josymar, ARIAS VALERO Juan Eduardo[4].

Keywords: BigData, knowledge discovery, supercomputing structures, SC3, large datasets processing.

Description: Thanks to major advances in information technology is concerned, the organizations have had to face a great challenge when analysing, discovering and understanding the proliferation of information from all processes internally and externally. This information can not be processed optimally for their traditional computing tools and they can be up to hundreds of petabytes. So new tools adapted to the processing of these large data volumes require high speed response, parallel to rapid obtaining results that are accurate and precise. These are the characteristics major opportunity for Big Data.

The Industrial University of Santander available infrastructure computing administered in part by the research group SC3, which is integrated through a platform called ”GridUIS-2”. ”GridUIS-2” integrates and connects different resources that are distributed in the central campus and headquarters Guatiguará. Among these technology resources there is the supercomputer (guane1) giving a service of high performance computing for the university community and scientific.

Considering the great willingness of these technological resources in the Industrial University of Santander, born the initiative to want to integrate and adapt these new computing tools to SC3, and thus have an important pillar in terms of information technology and supercomputing refers, as is big data, for the treatment of large volumes of data.

[3]Bachelor Thesis Research Paper mode.

[4]Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: PhD Raúl Ramos Pollán

INTRODUCCIÓN

Gracias a los grandes avances en cuanto a tecnologías de la información se refiere, las organizaciones han tenido que afrontar un gran desafió al momento de analizar, descubrir y entender la gran proliferación de información proveniente de todos los procesos realizados interna y externamente. Esta información no puede ser procesada óptimamente por sus herramientas tradicionales de computo y además pueden llegar a ser de cientos de petabytes. De manera que las nuevas herramientas adaptadas al procesamiento de estos grandes volúmenes de datos requieren de una gran velocidad de respuesta, paralelo a una rápida obtención de resultados que sean correctos y precisos. Estas son las características principales de oportunidad para Big Data.

Herramientas como Apache Hadoop, que utilizan nuevos paradigmas en función del constante flujo de la información. Apache Hadoop es un framework de código abierto para el almacenamiento distribuido y procesamiento de grandes conjuntos de datos sobre hardware comercial. Hadoop permite a las organizaciones obtener rápidamente una visión de cantidades masivas de datos estructurados y no estructurados.

La Universidad Industrial de Santander dispone de una infraestructura de computo administrada en parte por el grupo de investigación SC3, la cual se encuentra integrada a través de una plataforma denominada “GridUIS-2”. “GridUIS-2” integra y conecta diferentes recursos que se encuentran distribuidos en el campus central y la sede Guatiguará. Entre estos recursos tecnológicos se encuentra un supercomputador (guane1), dando así un servicio de computación de alto rendimiento a la comunidad universitaria y científica.

Teniendo en cuenta la gran disposición de estos recursos tecnológicos en la Universidad Industrial de Santander, nace la iniciativa al querer integrar y adaptar estas nuevas herramientas de computo al SC3, y así disponer de un pilar importante en cuanto a tecnologías de la información y supercomputación se refiere, como lo es Big Data, para el tratamiento de grandes volúmenes de datos.

1 DESCRIPCIÓN DEL PROYECTO

1.1 JUSTIFICACIÓN

La reciente proliferación de tecnologías y sistemas de información y grandes volúmenes de datos generados por ellas, requiere procesos de extracción y descubrimiento del conocimiento latente para su uso por expertos y usuarios en general. Es a través de numerosas herramientas tecnológicas y conceptuales que podemos llevar a cabo estos procesos basados en técnicas avanzadas de aprendizaje computacional y sistemas computacionales altamente escalables. Muchas de estas herramientas todavía son experimentales y su utilización e integración en infraestructuras y flujos de trabajo ya existentes no es trivial.

De aquí surge la necesidad de diseñar una arquitectura enfocada al tratamiento de estos grandes volúmenes de datos, que integre este tipo de herramientas y ofrezca un valor adicional para ser usada sobre una plataforma de computación de alto rendimiento como lo es el SC3. Esto dotaría a este tipo de laboratorios y, en especial a la Escuela de Ingeniería de Sistemas y a la UIS con la posibilidad de abordar problemas de tipo Big Data en distintas áreas de conocimiento.

1.2 DESCRIPCIÓN DEL PROBLEMA

El grupo de investigación SC3 dispone de una infraestructura de computación de alto rendimiento la cual puede resultar bastante útil a la hora abordar temas relacionados con el procesamiento de información a gran escala en tiempos relativamente bajos, temas relacionados con Big data. El hecho de que esta infraestructura no posea un servicio disponible para abordar este tipo de problemas, genera la gran iniciativa al querer integrar estas herramientas teniendo como base el supercomputador (guane1) y de esta manera aprovechar estos grandes recursos de computo.

Este servicio estaría a disposición de la Universidad Industrial de Santander pero de igual manera puede ser ofrecido a empresas y organizaciones que requieran el procesamiento de su información y que de alguna manera no disponen de las herramientas ni de la adecuada infraestructura para dicho procesamiento.

1.3 OBJETIVOS

1.3.1 Objetivo General

Diseñar una arquitectura de componentes software e implementar un piloto de caso de uso que integre herramientas de aprendizaje computacional y almacenamiento a gran escala, de despliegue y configuración automática en infraestructuras de cómputo del tipo del SC3.

1.3.2 Objetivos Específicos

- Desarrollar un prototipo de un servicio PaaS de Hadoop y Mahout sobre OAR / Kdeploy sobre SC3.
- Desarrollar un prototipo de un servicio PaaS del sistema Spark para computación en clúster con OAR / Kdeploy sobre SC3.
- Definir e implementar un caso de uso de analítica de datos sobre Hadoop / Mahout.
- Definir e implementar un caso de uso de analítica predictiva sobre Spark.
- Evaluar los casos de uso anteriores sobre los despliegues de Hadoop y Spark en la infraestructura de SC3 en términos de desempeño y escalabilidad.

2 MARCO TEÓRICO

2.1 HPC - COMPUTACIÓN DE ALTO RENDIMIENTO

La Computación de Alto Rendimiento o HPC (High Performance Computing), es la solución tecnológica desarrollada a partir de la demanda que generaron las aplicaciones

computacionales que requieren un alto nivel de procesamiento para obtener resultados en el menor tiempo posible. Usualmente las aplicaciones que utilizan esta clase de tecnología (HPC) para obtener resultados, son aplicaciones de desarrollo científico que se encuentran en fase de desarrollo y prueba, elaboradas por grupos de investigación u organizaciones enfocadas a diferentes ramas científicas, aplicaciones .

Desde el punto de vista del Hardware, HPC está conformado por tres tipos de infraestructuras [1]:

- Clúster: Grupo de cientos o miles de servidores con alta capacidad de procesamiento y de almacenamiento de datos, ubicados en un mismo lugar, los cuales trabajan juntos para dar solución a un mismo problema. Antiguamente, los supercomputadores eran máquinas de un solo procesador especial, ahora estos son a su vez catalogados como clústers.

- Cloud Computing: Una nube HPC proporciona el acceso a los recursos de forma escalable y dinámica, empleando la web como medio de comunicación y utiliza como base el modelo de computación como servicio (a-as-service).

- Grid Computing: Grupo de servidores distribuidos que trabajan en conjunto para ofrecer mayores recursos a las organizaciones que la conforman.

2.1.1 Clúster

El término clúster [2] se aplica a los conjuntos o conglomerados de computadoras construidos mediante la utilización de hardwares comunes y que se comportan como si fuesen una única computadora.

La tecnología de clústers ha evolucionado en apoyo de actividades que van desde aplicaciones de supercómputo y software de misiones críticas, servidores web y comercio electrónico, hasta bases de datos de alto rendimiento, entre otros usos.

El cómputo con clústers surge como resultado de la convergencia de varias tendencias actuales que incluyen la disponibilidad de microprocesadores económicos de alto rendimiento y redes de alta velocidad, el desarrollo de herramientas de software para cómputo distribuido de alto rendimiento, así como la creciente necesidad de potencia computacional para aplicaciones que la requieran.

Simplemente, un clúster es un grupo de múltiples ordenadores unidos mediante una red de alta velocidad, de tal forma que el conjunto es visto como un único ordenador, más potente que los comunes de escritorio.

Los clústers son usualmente empleados para mejorar el rendimiento y/o la disponibilidad por encima de la que es provista por un solo computador típicamente siendo más económico que computadores individuales de rapidez y disponibilidad comparables.

2.1.1.1 Características

Nodo Principal o Maestro

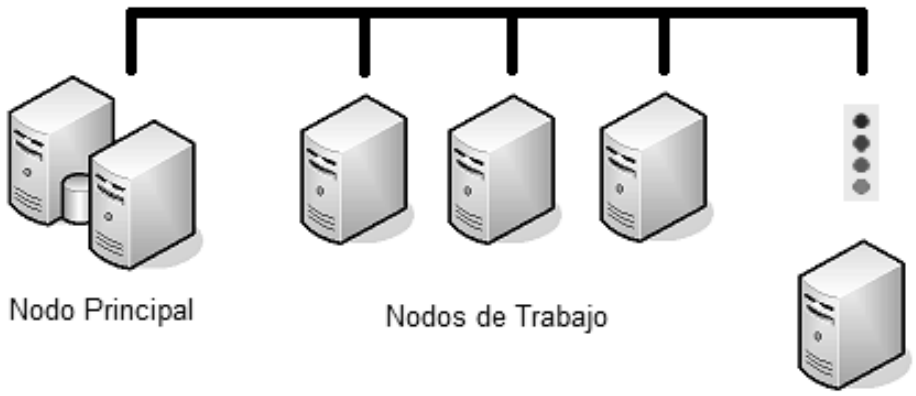
Es la puerta de entrada a una red compartida, por la cual se puede conectar al clúster. El nodo principal, tiene una o más redes por medio de las cuales se comunica a los nodos de trabajo, estas redes son privadas y solo se puede acceder a ellas a través del clúster. Contiene gran cantidad de almacenamiento que es compartida a través de la red por todos los nodos de trabajo.

Nodos de Trabajo

Son los nodos encargados de realizar el trabajo de la computación. Estos nodos son homogéneos, es decir, que son casi siempre idénticos en hardware y software, y se comunican entre sí y con el nodo principal en las redes privadas. La cantidad de nodos de trabajo en un clúster es variable pero puede ser una cantidad muy grande.

Para mantener ocupados todos los nodos, se necesita una buena conexión entre los dispositivos, la cual se clasifica según su Latencia, que corresponde al menor tiempo en el que un byte se puede enviar, y Ancho de Banda (Máxima velocidad de datos).

Figura 1: Clúster



Fuente: Autores.

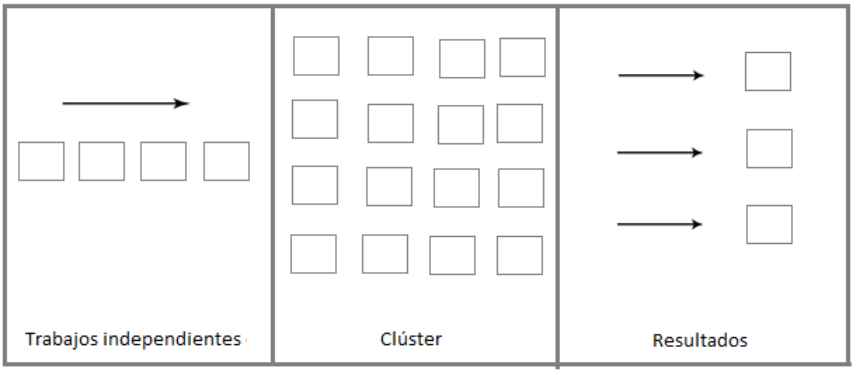
2.1.2 Clúster de Alto Rendimiento

Un clúster de alto rendimiento, esta especialmente adaptado para analizar y procesar problemas grandes y complejos que requieren de gran cantidad de potencia computacional,

para los cuales el tiempo de solución constituye un papel importante. Está formado por un número de máquinas individuales que se comunican y trabajan como si fuesen una sola maquina muy potente.

Un clúster de alto rendimiento puede trabajar de varias maneras, una de estas maneras, en forma de “Telaraña”, ya que un trabajo determinado no se ejecuta por un solo nodo de trabajo, en lugar de esto, es enviado a todo el clúster para realizar su ejecución. El Clúster administra los recursos necesarios para ejecutar un determinado trabajo y asigna los trabajos a un puesto en la cola de trabajos. Dependiendo de los recursos, todos los trabajos independientes, es decir los trabajos en los que no existe comunicación entre la ejecución de cada uno de ellos, pueden ejecutarse simultáneamente, aunque algunos esperan en la cola mientras que otros finalizan su ejecución. Este tipo de computación es *Computación Local a un nodo de clúster* [3], lo que significa que el nodo no se comunica con otros nodos, pero puede necesitar el acceso al sistema de archivos.

Figura 2: Computación Local a un Nodo del clúster



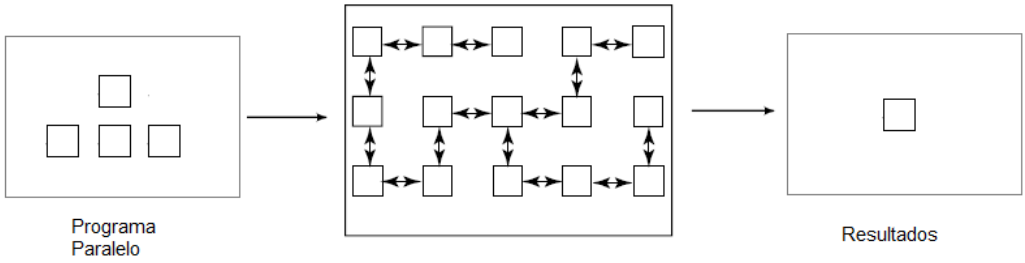
Fuente: Autores.

Otra forma de ejecución del clúster de alto rendimiento, consiste en que el clúster divide un trabajo HPC en varios sub-trabajos de un tamaño menor y ejecuta estos pequeños sub-trabajos en diferentes nodos. Este proceso de división se realiza a nivel se software. Debido a que los sub-trabajos necesitan comunicarse entre ellos, se puede causar tráfico de cómputo en el clúster. En este tipo de ejecución, es común utilizar miles de nodos para obtener el resultado de un solo trabajo, esto se conoce como Computación Paralela, lo que significa que un programa paralelo (es decir, un programa en el que se puede utilizar un software para dividirlo) solicita al clúster los recursos necesarios para ejecutarse, una vez realizada la división del programa y los recursos estén libres, el clúster ejecutará el programa utilizando varios nodos para la solución del mismo.

Existen dos tipos de Computación Paralela: Paralelismo de datos, el cual consiste en la división de un programa en subdominios; donde cada nodo trabaja sobre datos independientes, y Paralelismo de Tareas, en este tipo de computación se identifican las regiones o tareas del programa y solo se ejecutan en paralelo aquellas que son independientes.

Los tipos de memoria de la computación en Paralelo, son: Memoria compartida, en donde los trabajos acceden a la memoria global, y Memoria Distribuida, la cual utiliza comúnmente paso de mensajes para enviar y recibir datos y sincronizar procesos.

Figura 3: Computación Paralela



Fuente: Autores.

2.1.3 Cloud Computing - Computación en la nube

La computación en la nube o Cloud Computing [4] es un término general que involucra la entrega de servicios hospedados sobre internet. Estos servicios son divididos en general en tres categorías: Infraestructura como Servicio (IaaS), Plataforma como Servicio (PaaS), Software como Servicio (SaaS).

Un servicio en la nube tiene tres características distintas que se diferencian del hosting tradicional. Se vende sobre la demanda, típicamente por minutos o por horas; un usuario puede solicitar tanto como necesite de un servicio en cualquier momento; el servicio es totalmente administrado por el proveedor (el consumidor únicamente necesita un computador personal y acceso a internet). Significantes innovaciones en virtualización y computación distribuida, así como la mejora del acceso a internet de gran velocidad y una economía débil ha acelerado el interés en computación en la nube.

2.1.3.1 Infraestructura como Servicio (IaaS) Es un modelo de prestación en el cual una organización externaliza el equipo usado para soportar operaciones, incluyendo almacenamiento, hardware, servidores y componentes de red. El proveedor del servicio es dueño del equipo y es responsable por almacenar, operar y mantenerlo. Los servicios web de Amazon proveen una instancia de servicio virtual para iniciar, detener, acceder y configurar su almacenamiento y servidores. En la empresa, la computación en la nube permite a la compañía pagar únicamente por la capacidad que se necesita. Debido a que este modelo (paga por lo que uses) se asemeja a la forma en que la electricidad, el combustible y el agua son consumidos, a veces es llamado como utilidad de computación.

2.1.3.2 Plataforma como Servicio (PaaS) Es una manera de alquilar hardware, sistemas operativos, almacenamiento y capacidad de red sobre internet. El modelo de

entrega permite al consumidor alquilar servidores virtualizados y servicios asociados para ejecutar aplicaciones existentes o desarrollo y pruebas de nuevas aplicaciones.

PaaS tiene varias ventajas para desarrolladores. Con PaaS, características de los sistemas operativos pueden ser cambiados y actualizados frecuentemente. Grupos de desarrolladores geográficamente distribuidos pueden trabajar en proyectos de desarrollo. El servicio puede ser obtenido de diversas fuentes que cruzan límites internacionales. Los valores iniciales y actuales pueden ser reducidos usando servicios de infraestructura de un solo vendedor en vez de mantener múltiples instalaciones de hardware que a menudo realizan funciones duplicadas o sufren de problemas de incompatibilidad. Los gastos generales pueden ser minimizados por unificación de esfuerzos de desarrollo. Como parte negativa, PaaS sugiere algún riesgo de "lock-in" si el servicio requiere interfaces de servicio propietario o desarrollo de lenguajes. Otro potencial escollo es que la flexibilidad del servicio puede no conocer las necesidades de algunos usuarios cuyos requisitos evolucionan rápidamente.

2.1.3.3 Software como Servicio (SaaS)

Es un modelo de distribución de software en la cual las aplicaciones son hospedadas por un vendedor o proveedor de servicio que pone a disposición de los clientes a través una red, típicamente internet.

SaaS se está convirtiendo cada vez más en un modelo de entrega prevalente como una tecnología subyacente que soporta servicios web, arquitectura orientada al servicio (SOA) y nuevos enfoques de desarrollo, como Ajax. Mientras tanto, el servicio de banda ancha se hace cada vez más disponible para soportar acceso a usuarios desde más lugares alrededor del mundo.

SaaS está muy cercanamente relacionado con el ASP (Proveedor de Servicio de Aplicaciones) y con la demanda de modelos de entrega de software de computación. IDC identifica visiblemente dos modelos diferentes de entrega para SaaS. El modelo de administración de aplicación hospedada es similar a ASP: un proveedor de host comercialmente habilita el software para los clientes y lo entrega a través de la web. En el modelo de software por demanda, el proveedor ofrece a los clientes acceso basado en la red a una sola copia de una aplicación creada específicamente para distribución SaaS.

2.1.4 Grid Computing

El Grid Computing [5] o la Computación Grid, es una tecnología revolucionaria que utiliza recursos asociados a muchos ordenadores, redes y bases de datos, que se encuentran conectados a la red libremente, a los cuales se accede remotamente y que trabajan en conjunto para realizar tareas y problemas complejos, por lo general problemas de tipo científico o técnico que requiere un gran número de ciclos de procesamiento o acceso a grandes cantidades de datos.

En resumen, el Grid Computing es un conjunto de computadoras de diferentes instituciones conectadas en una red que utilizan sus recursos para trabajar en un solo problema al mismo tiempo.

Todos los recursos de un Grid necesitan una interconexión de hardware especial y un software que controle estos recursos para ensamblarlos en el Grid, ya que un Grid no está conformado por una sola organización utilizando sus recursos, sino que intervienen varias organizaciones geográficamente separadas cada una utilizando, compartiendo y virtualizando sus recursos computacionales. De esta forma, si una organización en un momento dado tiene todos sus recursos trabajando, y necesita aún más de los que dispone, es función del Grid equilibrar la utilización de sus recursos, buscando maquinas remotas, las cuales no estén trabajando y puedan ejecutar los trabajos de la organización que los necesita.

2.2 OAR

OAR [6] es un versátil administrador de recursos y tareas (también llamado planificador por lotes) para clústers de computación de alto rendimiento y otras infraestructuras computacionales.

Este sistema por lotes está cimentado en una base de datos (MySql o PostgreSQL), un lenguaje script (Perl) y una herramienta administrativa opcional escalable (componente del framework Taktuk). Está compuesto de módulos que interactúan sólo con base de datos y son ejecutados como programas independientes. Formalmente, no hay un API, el sistema es completamente por el esquema de la base de datos. Este enfoque facilita el desarrollo de módulos específicos. De hecho, cada módulo (como un planificador) puede ser desarrollado en cualquier lenguaje teniendo una librería de acceso a bases de datos.

2.3 KADEPLOY

Kadeploy [7] es un sistema de despliegue escalable, eficiente y seguro para computación clúster y grid. Provee un conjunto de herramientas para clonar, configurar (post-instalación) y administrar nodos en un clúster. Puede desplegar un clúster de 300 nodos en unos cuantos minutos, sin intervención del administrador del sistema. Puede desplegar Linux, BSD, Windows, Solaris.

Juega un papel clave en el banco de pruebas Grid'5000, donde permite que los usuarios reconfiguren el software del entorno en el nodo.

Kadeploy funciona de la siguiente manera:

- Configuración de entorno mínima**:** El nodo se reinicia en un entorno mínimo confiable que contiene todas las herramientas para el despliegue (herramientas de particionado, administrador de archivos, …) y el particionado requerido es realizado.

- Instalación del entorno: El entorno es emitido a todos los nodos y extraído en el disco. Algunas operaciones post-instalación pueden ser realizadas.

- Reiniciar en el entorno desplegado: Kadeploy toma como entrada un archivo que contiene el sistema operativo para desplegar, llamado entorno, y lo copia a los nodos destino. Como consecuencia, Kadeploy no instala un sistema operativo siguiendo un proceso de instalación clásico y el usuario tiene que proveer un archivo del sistema de archivos del sistema operativo (como un tarball, para entornos Linux).

2.4 MACHINE LEARNING

Machine Learning [8] Estudia algoritmos computacionales que aprenden a hacer distintas tareas. Podríamos, por ejemplo, estar interesados en aprender, para completar un trabajo, hacer predicciones precisas, o un comportamiento inteligente. El aprendizaje que se ha realizado siempre está basado en algún tipo de observaciones o datos, como ejemplos, experiencia directa o instrucción. Entonces, Machine Learning hace referencia al aprendizaje como herramienta para hacer mejor las cosas en el futuro basado en lo que fue experimentado en el pasado.

El énfasis de Machine Learning está en métodos automáticos. En otras palabras, el objetivo es idear algoritmos de aprendizaje que hagan el proceso automáticamente sin intervención o asistencia humana. El paradigma de Machine Learning puede ser visto como “programación por ejemplo”.

A menudo tenemos una tarea específica en mente, como el filtrado de spam. En lugar de programar el computador para que resuelva el problema directamente, en Machine Learning, se buscan métodos por los cuales el computador arroje su propio programa basado en ejemplos que nosotros proveemos.

Machine Learning es una sub-área muy importante de la inteligencia artificial. Es muy poco probable que seamos capaces de construir algún tipo de sistema inteligente que tenga la suficiente capacidad para soportar alguna de las facilidades que asociamos con inteligencia, como el lenguaje o la visión, sin utilizar el aprendizaje para lograr dicho objetivo. Estas tareas son, de otro modo, simplemente muy difíciles de resolver.

Además, no consideraremos un sistema verdaderamente inteligente si fuese incapaz de aprender, desde que aprender es la esencia misma de la inteligencia. Además de una sub-área de la IA, Machine Learning se encuentra ampliamente intersectada con otros campos especialmente estadística, matemática, física, teoría de la computación, ciencia y muchos otros.

2.5 BIG DATA

Es un nuevo enfoque de entendimiento y toma de decisiones el cual es utilizado para describir grandes cantidades de datos bien sean, estructurados, no estructurados o semi-estructurados, que serían muy costosos y tomarían demasiado tiempo en cargarlos en bases de datos tradicionales. En este sentido, el concepto de Big Data [9] aplica a todos aquellos

datos que no pueden ser procesados y/o analizados para extraer conocimiento de manera tradicional. Por otra parte, Big Data no se refiere a una cantidad en específico de datos, normalmente esta técnica se utiliza cuando se habla de petabytes y exabytes.

Actualmente los seres humanos están generando cada vez más cantidades gigantescas de datos, las empresas guardan información de sus proveedores, preferencias de clientes, datos transaccionales, etc., solamente hablando del sector privado. Los gobiernos guardan información de censos, registros, datos biométricos, impuestos, historias médicas, etc., si se añaden los datos generados por ubicación geográfica, las redes sociales (actualmente se generan 100000 tweets por minuto y los usuarios de Facebook comparten 684478 piezas de contenido por minuto), y en general todos datos generados por los dispositivos móviles, se habla de que se generan en el mundo cerca de 2.5 quintillones de bytes diariamente.

$$1quintillón = 1^{30} = 1000000000000000000000000000000$$

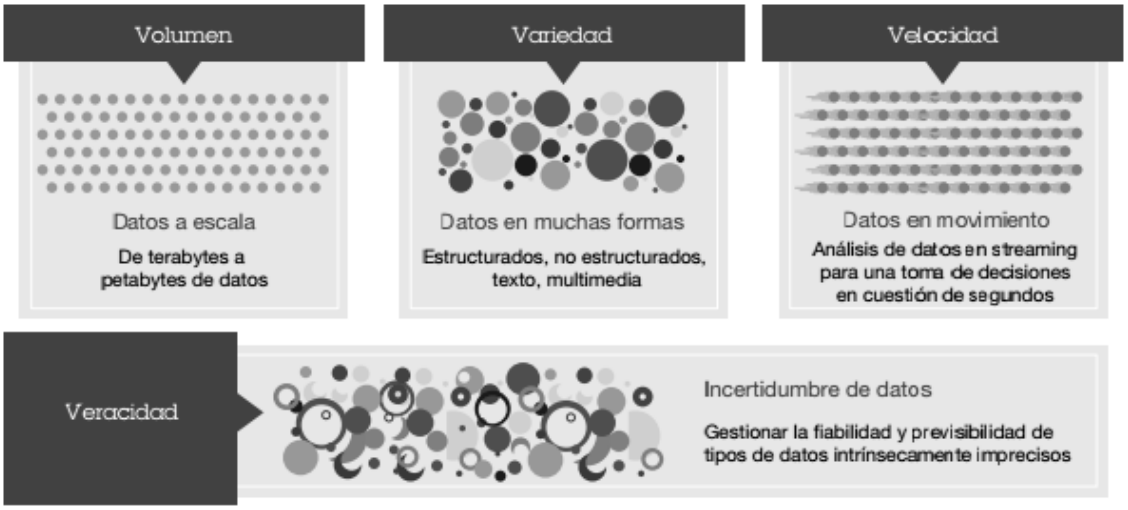
Toda esta información corresponde únicamente a los datos generados por los seres humanos, también contribuyen al crecimiento de datos las comunicaciones M2M (machine-to-machine) cuyo valor en la creación de grandes cantidades de datos es muy significativa.

2.5.1 Las cuatro dimensiones de Big Data

Cuando se habla de Big Data, se habla de una combinación de características, que definidas de manera apropiada, pueden crear oportunidades enfocadas a la obtención de ventajas competitivas en empresas y organizaciones, relacionadas actualmente con el mercado digitalizado, ya que en todos los sectores existe la posibilidad de utilizar las nuevas tecnologías y analíticas de Big Data para mejorar la toma de decisiones y el rendimiento.

De esta manera se coincide con una forma útil de caracterizar tres dimensiones de Big Data, “las tres V” [10]: Volumen, Variedad y Velocidad. y si bien estas dimensiones engloban los principales atributos de Big Data, creemos que las empresas deben tener en cuenta una cuarta e importante dimensión: la veracidad. Incluir la veracidad como el cuarto atributo de Big Data pone de relieve la importancia de abordar y gestionar la incertidumbre inherente a algunos tipos de datos.

Figura 4: Las Cuatro Dimensiones de Big Data



Fuente: IBM Global Business Services Business Analytics and Optimisation, Analytics: el uso de big data en el mundo real

2.5.1.1 Volumen La cantidad de datos. Siendo quizá la característica que se asocia con mayor frecuencia a Big Data, el volumen hace referencia a las cantidades masivas de datos que las organizaciones intentan aprovechar para mejorar la toma de decisiones en toda la empresa u organización. Los volúmenes de datos continúan aumentado a un ritmo sin precedentes, ya que pasaron de ser generados por las propias personas, a ser generados automáticamente por maquinas, redes e interacciones personales en sistemas como las redes sociales.

2.5.1.2 Variedad Diferentes tipos y fuentes de datos. La variedad tiene que ver con gestionar la complejidad de múltiples tipos de datos, incluidos los datos estructurados, semiestructurados y no estructurados. Las organizaciones necesitan integrar y analizar datos de un complejo abanico de fuentes de información tanto tradicional como no tradicional procedentes tanto de dentro como de fuera de la empresa. Con la profusión de sensores, dispositivos inteligentes y tecnologías de colaboración social, los datos que se generan presentan innumerables formas entre las que se incluyen texto, datos web, tweets, datos de sensores, audio, vídeo, secuencias de clic, archivos de registro y mucho más.

2.5.1.3 Velocidad Los datos en movimiento. La velocidad a la que se crean, procesan y analizan los datos continúa aumentando. Contribuir a una mayor velocidad es la naturaleza en tiempo real de la creación de datos, así como la necesidad de incorporar datos en streaming a los procesos de negocio y la toma de decisiones. La velocidad afecta a la latencia: el tiempo de espera entre el momento en el que se crean los datos, el momento en el que se captan y el momento en el que están accesibles. Hoy en día, los datos se generan de forma continua a una velocidad a la que a los sistemas tradicionales les resulta imposible captarlos, almacenarlos y analizarlos. Para los procesos en los que el tiempo

resulta fundamental, tales como la detección de fraude en tiempo real o el marketing "instantáneo" multicanal, ciertos tipos de datos deben analizarse en tiempo real para que resulten útiles para cualquier organización.

2.5.1.4 Veracidad La incertidumbre de los datos. La veracidad hace referencia al nivel de fiabilidad asociado a ciertos tipos de datos. Esforzarse por conseguir unos datos de alta calidad es un requisito importante y un reto fundamental de Big Data, pero incluso los mejores métodos de limpieza de datos no pueden eliminar la imprevisibilidad inherente de algunos datos, como el tiempo, la economía o las futuras decisiones de una organización. La necesidad de reconocer y planificar la incertidumbre es una dimensión de Big Data que surge a medida que los directivos a cargo de las organizaciones intentan comprender mejor el mundo incierto que les rodea.

3 FRAMEWORKS PARA ANÁLISIS DE BIG DATA

3.1 HADOOP

Hadoop proporciona un framework, escrito en Java, sobre el cual se desarrollan aplicaciones distribuidas que requieren un uso intensivo de datos y de alta escalabilidad. Es un proyecto de la Apache Software Fundation [11] que aglutina diferentes subproyectos, donde se desarrolla software de código abierto.

Se presenta como una solución para los programadores sin experiencia en desarrollo de aplicaciones para entornos distribuidos, dado que oculta la implementación de detalles propios de estos sistemas: paralelización de tareas, administración de procesos, balanceo de carga y tolerancia a fallos.

Hadoop implementa, entre otras cosas, el paradigma MapReduce y un sistema de ficheros distribuido denominado HDFS (Hadoop Distributed File System) inspirado en las publicaciones de Google sobre el modelo de programación MapReduce y sobre su sistema de ficheros distribuido denominado GFS (Google File System).

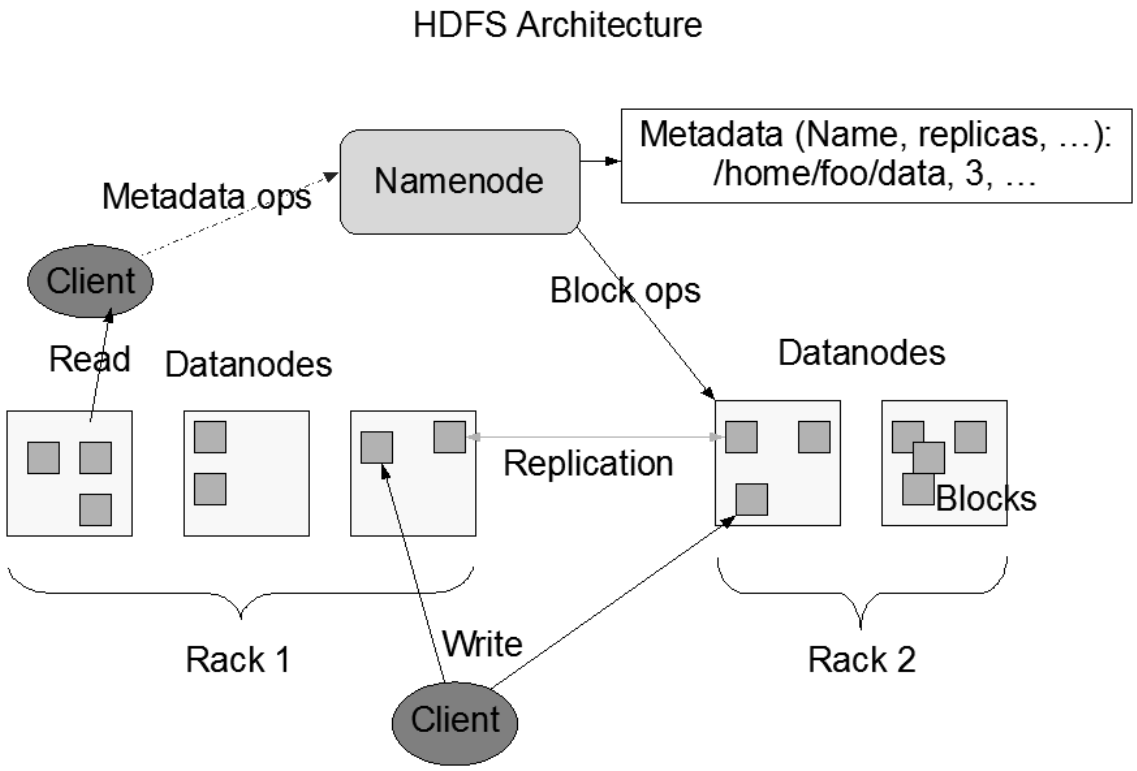
3.1.1 HDFS

El Sistema de Ficheros Distribuido de Hadoop (HDFS [12], siglas en ingles) es un sistema de archivos distribuido diseñado para almacenar grandes cantidades de información, del orden de terabytes o petabytes, y pensado para correr en hardware relativamente económico, ampliamente disponible y mas o menos intercambiable. Distribuye la información a manera de bloques, que son almacenados y guardados en los discos locales de los nodos pertenecientes al clúster. Tiene muchas similitudes con sistemas de archivos distribuidos existentes, sin embargo, las diferencias con otros sistemas de archivos

distribuidos son significativas. Resaltando como diferencia notable, que esta pensado para aplicaciones que siguen modelos de una sola escritura y muchas lecturas, minimizando el control de concurrencia, simplificando la coherencia de datos y proporcionando un acceso de alto rendimiento.

HDFS se compone de un grupo de nodos interconectados, donde residen los archivos y directorios. Presenta una arquitectura master/worker basada en un único nodo maestro, denominado NameNode, que maneja el espacio de nombres del sistema y regula el acceso de los clientes a los ficheros, redirigiéndolos a los nodos de datos que contienen la información, denominados DataNodes, que son los encargados de gestionar el almacenamiento en los discos locales del propio nodo. La Figura 5 muestra de forma gráfica la arquitectura HDFS.

Figura 5: Arquitectura HDFS



Fuente: HDFS Architecture Guide,
http://hadoop.apache.org/docs/current1/hdfs_design.html

Tanto el NameNode como los DataNodes son componentes software, diseñados para funcionar, de manera desacoplada, en máquinas genéricas a través de sistemas operativos heterogéneos. HDFS ha sido construido utilizando el lenguaje de programación Java, por lo tanto, cualquier máquina que soporte dicho lenguaje puede ejecutar HDFS. Una instalación típica consta de una máquina dedicada, donde se ejecutará el NameNode, y en cada una de las máquinas restantes que constituyen el clúster, se ejecutará un DataNode.

3.1.1.1 Caracteristicas de HDFS

Tolerancia a fallos de Hardware

Una instancia de HDFS puede consistir de cientos o miles de servidores, cada uno ordenando parte de los datos del sistema de archivos. El hecho de que hay un gran numero de componentes y que cada componente tiene tiene una probabilidad no trivial de fallar, significa que algún componente de HDFS es siempre no funcional. Por lo tanto, la detección de fallos y recuperación rápida y automática de ellos es un objetivo arquitectónico clave de HDFS.

Grandes conjuntos de datos

Las aplicaciones que corren en HDFS tienen grandes cantidades de datos. Un archivo típico en HDFS tiene gigabytes o terabytes en tamaño. Luego, HDFS se adapta para soportar grandes archivos. Debería escalar a cientos de nodos en un solo clúster y soportar decenas de millones de archivos en una sola instancia.

Modelo de coherencia simple

Las aplicaciones HDFS necesitan un modelo de acceso a archivos escribir una vez, leer muchas veces. Un archivo una vez creado, escrito y cerrado necesita ser cambiado. Esta presunción simplifica problemas de coherencia de datos y permite alto rendimiento de acceso a datos. Una aplicación MapReduce o un rastreador web cabe perfectamente dentro dentro de este modelo.

“Mover el computo es mas económico que mover los datos”

Un computo solicitado por una aplicación es mucho mas eficiente si es ejecutada cerca a los datos donde este opera. Esto es especialmente cierto cuando el tamaño del conjunto de datos es enorme. Esto minimiza la congestión el la red local e incrementa el rendimiento general del sistema. La presunción es que a menudo es mejor migrar la computación cerca a donde los datos están localizados en lugar de mover los datos donde la aplicación esta corriendo. HDFS provee interfaces para aplicaciones que les permite moverse mas cerca a donde los datos están localizados.

3.1.1.2 NameNode y Data Node Una aplicación cliente, que desea leer un fichero en HDFS, debe contactar primero con el NameNode, para determinar en lugar en el cual está almacenada la información que requiere. En respuesta al cliente, el NameNode retorna el identificador del bloque más relevante y el nodo en el cual está almacenado. A

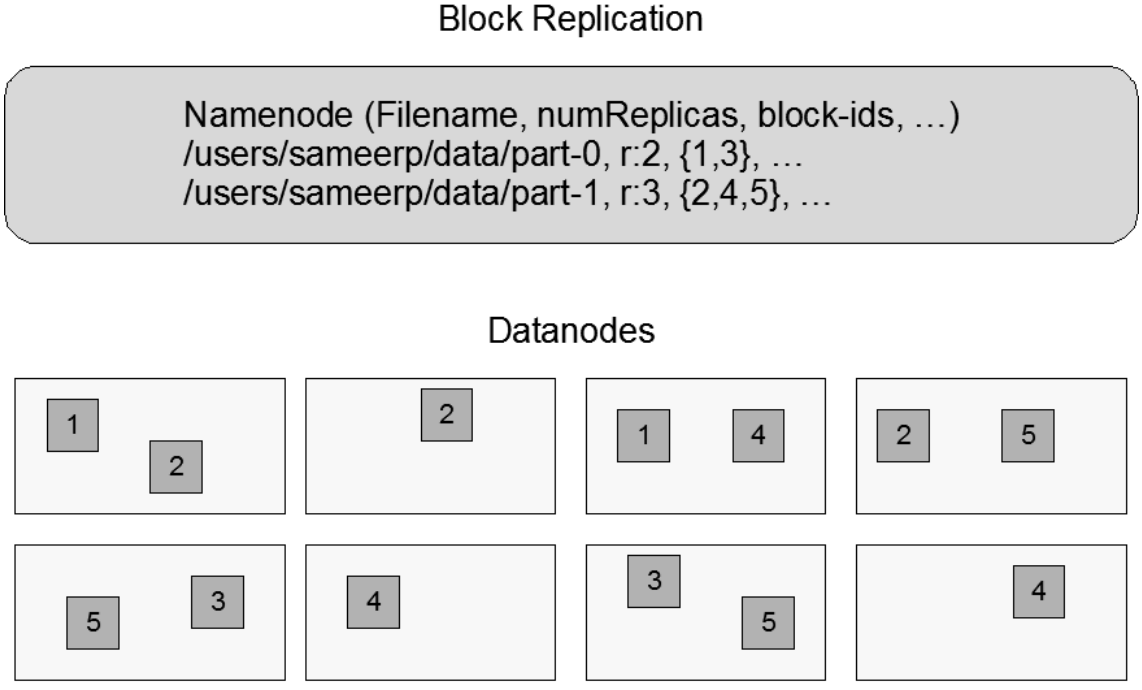
continuación, el cliente contacta con el DataNode para recuperar la información requerida. Los bloques se encuentran almacenados en el sistema de ficheros local de la máquina, y el HDFS se encuentra en la parte superior de la pila del sistema operativo estándar (por ejemplo Linux). Una característica importante del diseño de este sistema de ficheros es, que la información nunca se mueve al NameNode. Toda la transferencia de información se produce directamente entre los clientes y los nodos de datos. La comunicación con el NameNode sólo implica transferencia de meta-información.

La existencia de un solo NameNode en un cluster simplifica de gran manera la arquitectura del sistema. El NameNode es el arbitro y almacén de todo el metadata de HDFS. El sistema esta diseñado de tal manera que los datos de usuario nunca fluyen a través del NameNode.

3.1.1.3 Espacio de Nombres del sistema de archivos HDFS soporta una organización de archivos jerárquica tradicional. Un usuario o aplicación puede crear directorios y almacenar archivos adentro de estos directorios. La jerarquía del espacio de nombres es similar a la mayoría de otros sistemas de archivos existentes; se puede crear y eliminar archivos, mover un archivo de un directorio a otro, o renombrar un archivo.

El NameNode mantiene el espacio de nombres del sistema de archivos. Cualquier cambio al espacio de nombres o sus propiedades es grabado por el NameNode. Una aplicación puede especificar el numero de replicas de un archivo que debería ser mantenido por HDFS. El numero de copias de un archivo es llamado el factor de replica de un archivo. Esta información es almacenada por el NameNode.

Figura 6: Bloque de Replica



Fuente: HDFS Architecture Guide,
http://hadoop.apache.org/docs/current1/hdfs_design.html

Para que el sistema de ficheros sea tolerante a fallos, HDFS replica los bloques de ficheros. Una aplicación puede especificar el número de réplicas para un fichero en el momento que es creado, pudiendo ser cambiado en cualquier momento. El NameNode toma las decisiones relativas a la replicación de bloques.

Uno de los objetivos principales de HDFS es dar soporte a ficheros de gran tamaño. El tamaño típico de un bloque de fichero en HDFS es 64Mb o 128Mb. Un fichero está compuesto de uno o varios bloques de 64/128Mb, y HDFS trata de colocar cada bloque en nodos de datos separados, distribuyendo la información a lo largo del clúster.

Los bloques no siempre pueden ser colocados de manera uniforme en los nodos de datos, lo que significa que el espacio disponible por uno o más nodos de datos puede estar infrautilizado. Otro caso común, que provoca que la distribución de los datos entre los diferentes nodos no esté balanceada, es la adición de nodos de datos al clúster. HDFS proporciona rebalanceo de bloques de datos utilizando diferentes modelos. Un modelo permite mover los bloques de un nodo de datos a otro, de forma automática, si el espacio libre en un nodo cae demasiado. Otro modelo permite crear, dinámicamente, réplicas adicionales para un determinado fichero, si se produce un aumento repentino de la demanda, rebalanceando otros bloques en el clúster. HDFS también proporciona comandos que permite realizar tareas de reajuste de forma manual.

3.1.2 Hadoop MapReduce

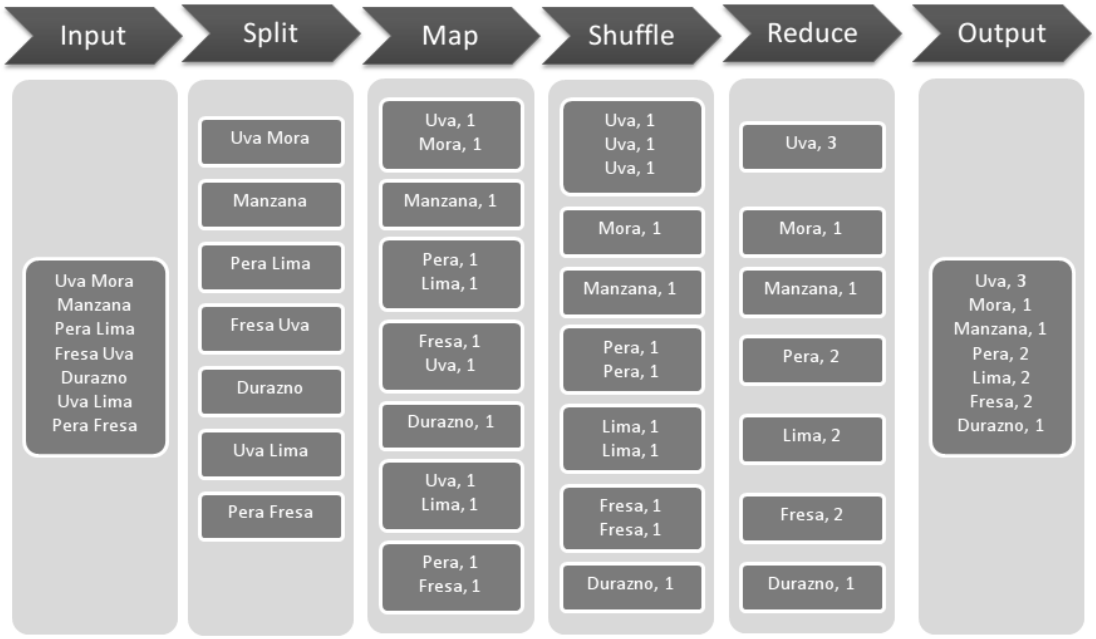
Hadoop proporciona un entorno de ejecución orientado a aplicaciones desarrolladas bajo el modelo de programación MapReduce [13]. Bajo este modelo, la ejecución de una aplicación presenta dos etapas:

3.1.2.1 Map Donde se realiza la ingestión y la transformación de los datos de entrada, en la cual los registros de entrada pueden ser son procesados en paralelo.

3.1.2.2 Reduce Fase de agregación o resumen, donde todos los registros asociados entre sí deben ser procesados juntos por una misma entidad.

La idea principal sobre la cual gira el entorno de ejecución Hadoop MapReduce es, que la entrada puede ser dividida en fragmentos (split), y cada fragmento, puede ser tratado de forma independiente por una tarea map. Los resultados de procesar cada fragmento, pueden ser físicamente divididos en grupos distintos. Cada grupo se ordena (shuffle) y se pasa a una tarea reduce.

Figura 7: Ejemplo de las Etapas de MapReduce

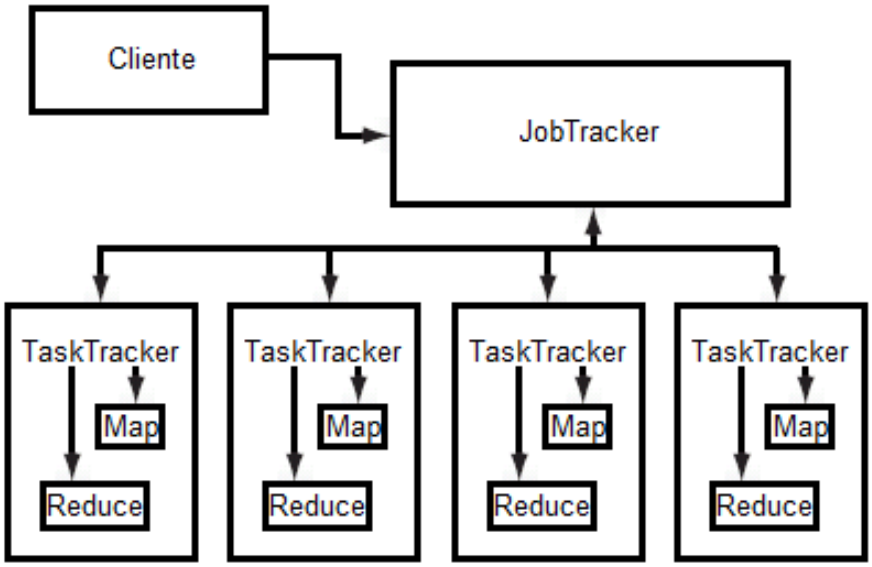


Fuente: Autores.

Una tarea map puede ejecutarse en cualquier nodo de cómputo del clúster, y múltiples tareas map pueden ejecutarse en paralelo en el clúster. La tarea map es responsable de transformar los registros de entrada en duplas <clave, valor>. La salida de todos los map se dividirá en particiones, y cada partición será ordenada por clave. Habrá una partición por cada tarea reduce. Tanto la clave como los valores asociados a la clave son procesados por la tarea reduce, siendo posible que varias tareas reduce se ejecuten en paralelo.

3.1.2.3 JobTracker y TaskTracker El entorno de ejecución Hadoop Mapreduce está formado por dos componentes principales: JobTracker y TaskTracker, ambos codificados en Java. Al igual que en el HDFS, el entorno de ejecución presenta una arquitectura cliente/servidor. En un clúster hay un único JobTracker, siendo su labor principal la gestión de los TaskTrackers, entre los que distribuye los trabajos MapReduce que recibe. Los TaskTrackers son los encargados de ejecutar las tareas map/reduce. En un clúster típico, se ejecuta un TaskTracker por nodo de cómputo. En la Figura 8 se muestra de forma esquemática la interacción entre JobTracker y TaskTracker.

Figura 8: Interacción JobTracker y TaskTracker



Fuente: Autores.

El JobTracker es el enlace entre la aplicación y Hadoop. Una vez se envía un trabajo al clúster, el JobTracker determina el plan de ejecución en base a los ficheros a procesar, asigna nodos de cómputo a las diferentes tareas, y supervisa todas las tareas que se están ejecutando. En el caso de fallar una tarea, el JobTracker relanza la tarea, posiblemente en un nodo diferente, existiendo un límite predefinido de intentos en el caso que dicha tarea falle de forma reiterada. Como se ha comentado anteriormente, sólo se ejecuta un JobTracker por clúster Hadoop.

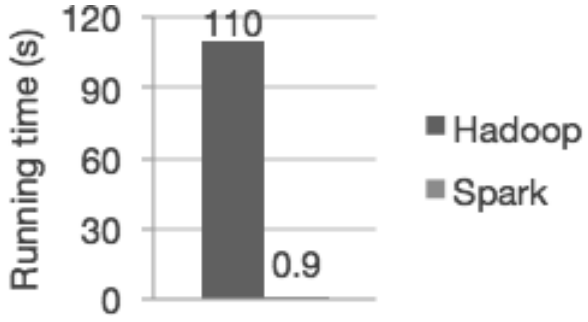
Cada TaskTracker es responsable de ejecutar las tareas que el JobTracker le ha asignado. Cada TaskTracker puede ejecutar varias tareas map/reduce en paralelo. El JobTracker crea instancias Java separadas para la ejecución de cada tarea. De esa manera se garantiza que, en caso de fallar la ejecución de una tarea, no afecta al resto de tareas ni tampoco al propio JobTracker.

3.2 APACHE SPARK

Desarrollado en Scala, Spark [14] es una plataforma de computación de código abierto para análisis y procesos avanzados, que tiene muchas ventajas sobre Hadoop. Desde el principio, Spark fue diseñado para soportar en memoria algoritmos iterativos que se pudiesen desarrollar sin escribir un conjunto de resultados cada vez que se procesaba un dato. Esta habilidad para mantener todo en memoria es una técnica de computación de alto rendimiento aplicado al análisis avanzado, la cual permite que Spark tenga unas

velocidades de procesamiento que sean 100 veces más rápidas que las conseguidas utilizando MapReduce.

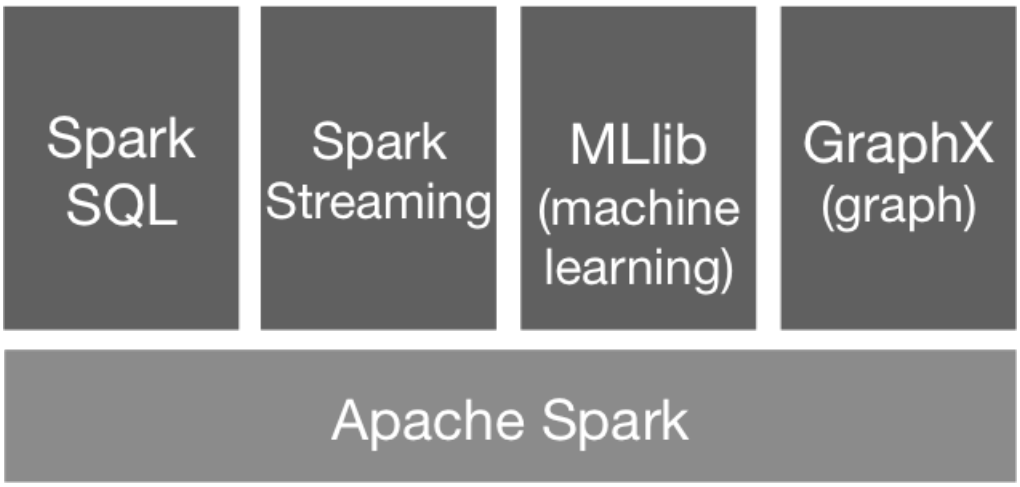
Figura 9: Logistic Regression en Hadoop y Spark



Fuente: http://spark.apache.org/

Spark tiene un Framework integrado para implementar análisis avanzados que incluye la librería MLlib [15], el motor gráfico GraphX [16], Spark Streaming [17], y la herramienta de consulta Shark [18]. Esta plataforma asegura a los usuarios la consistencia en los resultados a través de distintos tipos de análisis. Una de las cosas realmente buenas de Spark es que lo puedes descargar y ejecutar en un ordenador personal.

Figura 10: Herramientas Integradas en Apache Spark



Fuente: http://spark.apache.org/

3.3 APACHE AMBARI

Apache Ambari [19] [20] framework de código abierto que tiene como objetivo hacer más sencilla la gestión de Hadoop mediante el desarrollo de software para el aprovisionamiento, gestión y seguimiento de las agrupaciones Apache Hadoop. Ambari ofrece una interfaz web intuitiva y fácil de usar además que permite a los administradores del sistema:

Prestación de clúster Hadoop

Sin importar el tamaño del clúster, el despliegue y mantenimiento de los hosts se simplifica utilizando Ambari. Incluye una interfaz Web que permite la fácil prestación, configuración y testeo de los servicios y componentes principales de Hadoop.

Administración de clúster Hadoop

Ambari provee herramientas que simplifican la administración del clúster. La interfase web permite controlar el ciclo de vida de los servicios y componentes de Hadoop, modificar configuraciones y administrar el crecimiento de un clúster.

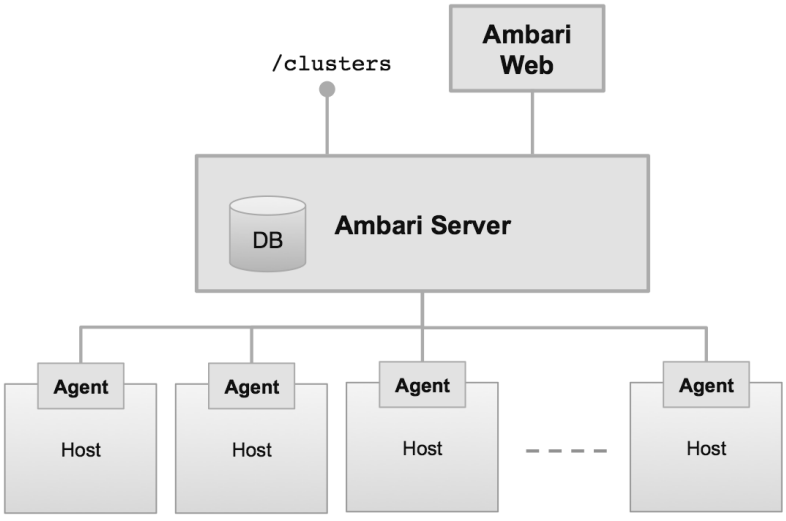
Monitoreo de clúster Haddop

Provee una vista instantánea a la salud del clúster. Ambari pre-configura alertas para observar los servicios Hadoop y visualizar datos operacionales del clúster en una interfaz web sencilla.

3.3.1 Arquitectura

El servidor Ambari funciona como un punto de recolección para datos a través del clúster. Cada host tiene una copia del agente Ambari el cual permite que el servidor Ambari controle cada host. Además cada host tiene una copia de Ganglia Monitor el cual reúne métricas de información que son pasadas a Ganglia Collector, y luego al servidor Ambari.

Figura 11: Arquitectura Ambari



Fuente: http://labitacoranet.wordpress.com/2013/10/01/monitorizacion-en-hadoop/

4 ESTADO DEL ARTE

4.1 INFRAESTRUCTURAS COMPUTACIONALES

La posibilidad ofrecida por los recursos computacionales hoy en día para el desarrollo de la ciencia y la ingeniería en nuevos campos está bien establecida. Simulación Basada en Ingeniería y Ciencia (SBE&S) hoy a alcanzado un nivel de capacidad predictiva que ahora firmemente complementa los pilares tradicionales de la teoría y la experimentación/observación. Como resultado, la simulación computacional es más fuerte y tiene más impacto hoy en día que en cualquier otra época en la historia de la humanidad. Muchas tecnologías críticas, incluyendo aquellas para desarrollar nuevas fuentes de energía y el cambio del factor costo-beneficio en el cuidado de la salud, están en el horizonte en el cual no pueden ser entendidas, desarrolladas, o utilizadas sin la simulación.

En particular, el reporte se enfatiza en el hecho que empoderar a una nación con capacidades HPC (High-Performance Computing) es clave para su desarrollo, pero ello requiere no solo acceso a recursos de computación integral, sino que hay una necesidad urgente de estar equipado con las herramientas apropiadas de software para cada problema científico y dominar y preparar tanto a científicos e ingenieros para utilizar HPC en colaboración.

Adicionalmente, los recursos computacionales se convierten rápidamente más baratos y disponibles, como se puede ser visto en la evolución de los supercomputadores que toman parte en el TOP 500, el cual cada seis meses lista a los 500 supercomputadores más poderosos del mundo.

Hasta el día de hoy, los supercomputadores que toman parte de la lista son cada vez más de naturaleza similar, más del 80% del TOP 500, son creados como arquitecturas clúster, interconectadas con Infiniband o Gigabit Ethernet, con procesadores de 64 bits de Intel o AMD, operados por sistemas operativos Linux. Esto significa que los centros de supercomputación se están convirtiendo cada vez más en una mercancía, siendo construidos con hardware muy ampliamente disponible. Es sencillo inferir que, fuera del TOP 500, que esta visión yace en la raíz de la actual proliferación de centros de supercomputación de todas las tallas alrededor del mundo.

Estos dos problemas (el uso de los recursos de computación como parte fundamental del desarrollo científico y de ingeniería y la proliferación de centros de supercomputación) han provocado muchos esfuerzos de todo tipo de instituciones en el proceso de integración de los recursos de computación distribuida y dispersa para construir “eInfraestructures” a través del mundo con el objetivo de habilitar a la ciencia y la ingeniería el uso de esta nueva herramienta como parte fundamental de sus desarrollos, sin la cual tal desarrollo simplemente no existiría.

Véase, por ejemplo los proyectos de infraestructura en los Programas Marco de la Unión Europea para el desarrollo y la Innovación; y la Fundación Nacional de los EE.UU para la Investigación y los Programas de Desarrollo.

Las eInfraestructures se ha hecho asequibles a través de cierto modelo de computación y método de acceso, cada uno direccionado a diferentes tipos de problemas, usuarios y basado en diferentes tecnologías sobresalientes, etc. Los supercomputadores a menudo corren aplicaciones compuestas de trabajos comunicativos a través de sistemas de colas por lotes. Las grid típicamente corren aplicaciones compuestas de trabajos no comunicativos a través de un middleware delegando el almacenaje y los recursos de computación.

La nube está basada en tecnologías de visualización y corren procesos bajo modelos de computación bien definidos de acuerdo a fin del objeto consumido por el usuario: Software como Servicio (SaaS), Plataforma como Servicio (PaaS) e Infraestructura como Servicio (IaaS). La nube es usada más frecuentemente por la industria, pero la academia está adoptando gradualmente sus modelos de computación.

4.1.1 Uso de HCP eINFRASTRUCTURES

Las grids han permanecido mayoritariamente en el área académica, mientras que la Nube parece estar ganado terreno en la industria. Sin embargo algunos problemas aún permanecen sin resolver en ambos mundos.

Uno de esos problemas es la interoperabilidad, el cual se refiere a la capacidad de diferentes eInfrastructures de intercambiar perfectamente recursos para cumplir con la demanda del usuario (tales como “jobs” de usuario siendo transferidos entre diferentes uniones de grids o máquinas virtuales entre diferentes nubes) y que permanecen aún sin resolver tanto en sistemas grid y nube.

Esto es principalmente un problema organizacional desde que, técnicamente, la interoperabilidad es un problema de estandarización. Varias arquitecturas, cuerpos de estandarización y grupos de trabajos han sido establecidos para buscar soluciones a estos problemas.

No obstante, su éxito depende en gran medida del grado de consenso logrado entre las diferentes partes interesadas en los sectores públicos y privados (proveedores de recursos, usuarios, proveedores de servicios, desarrolladores, etc.).

Otro punto es el costo de uso, en términos de esfuerzo requerido para aprovechar eficientemente las eInfrastructures. En las grid se refiere al porcentaje de trabajos que fallan debido al middleware y causas especificas no relacionadas con las aplicaciones (confianza del trabajo), y los sobre costos en la ejecución del trabajo debido al manejo del middleware (latencia del trabajo). Dependiendo del middleware y las instalaciones de centros de datos grid típicamente se muestran latencias en orden den varios minutos. Es también sabido que las tazas de fallo en los trabajos son significativamente altas que van desde el 5% hasta el 25% de fallas presentadas debido a razones del middleware, aunque esto está mejorando en los últimos lanzamientos.

Todo esto requiere que los usuarios y administradores desarrollen sus propias herramientas (para volver a entregar trabajos fallidos, revisión de la integridad de los datos, control de

trabajo, etc.) para asegurarse de que su producción científica este bien ejecutada y que la infraestructura pueda ser administrada, elevando considerablemente el costo (esfuerzo) de usar y mantener las grids, por consiguiente desacelerando su adopción en un sentido amplio.

4.2 BIG DATA

Hace referencia a conjuntos de datos los cuales están mas allá de la habilidad de las típicas herramientas de base de datos para capturar, administrar, y analizar datos. En esta definición se entiende que como capacidades tecnológicas avanzan en el tiempo, los tamaños de los conjuntos de datos que califican como Big Data también aumentaran, inclusive a una tasa mayor.

De hecho, la importancia de Big Data y cómo se ha convertido en un punto clave en futuros desarrollos tecnológicos está ilustrado por la reciente Investigación Big Data e Iniciativas de Desarrollo de los US, donde la administración Obama anunció que los Estados Unidos está invirtiendo $200 millones para mejorar la habilidad del gobierno de “extraer conocimiento e ideas de una gran y compleja colección de datos digitales,” en un esfuerzo por acelerar descubrimientos nacionales en ciencia e ingeniería.

El manejo de Big Data no puede ser emprendido utilizando bases de datos tradicionales y paquetes de estadísticas/visualización de escritorio clásicos, requiriendo en su lugar software paralelo masivo corriendo en decenas, cientos, o inclusive miles de servidores.

Los retos con Big Data son como explotar la variedad de fuentes de datos, como escalar cuando la cantidad de datos crece rápidamente, y como administrar y utilizar eficientemente el gran volumen de contenido de cada dato y la colección completa cuando los tamaños crecen sobre terabytes, exabytes o zettabytes.

Estos son los retos pero Big Data es más que esto, es una oportunidad de atacar problemas clásicos no resueltos de otra manera, explotando el conocimiento contenido en grandes bases de datos, la biomedicina no es ajena a esta tendencia, de hecho uno de los mejores campos para trabajar con Big Data es la investigación biomédica y la asistencia médica.

Las dificultades en el manejo de Big Data han sido gradualmente superadas por nuevos modelos computacionales y nuevas tecnologías, notablemente, el modelo Google’s Big Table, sistemas inspirados en bases de datos NO-SQL (HBase, Cassandra, DynamoDB) para el almacenaje; y Hadoop para computación. En general, la expresividad presente en las bases de datos relacionales es abandonada por modelos de datos simples, bien ajustados a la naturaleza de las aplicaciones, explotadas por tecnologías para hacerlas trivialmente escalables, esto es, rendimiento aumentado (escalabilidad) es logrado simplemente arrojando nuevo hardware básico. En el mismo sentido, desde un punto de vista computacional, los algoritmos han sido rediseñados para ser compuestos de tareas independientes no comunicativas de modo que la aceleración aumentada puede ser lograda simplemente desplegando componentes computaciones adicionales.

5 METODOLOGIA

5.1 Estrategia general

Se probarán los esquemas de configuración e instalación sobre máquinas virtuales Vbox, esto para economizar tiempo de SC3. Logísticamente es más fácil, el ciclo de desarrollo-prueba es más corto y además:

- Desarrollar imágenes de despliege OAR/Kadeploy sobre SC3
- Definir métricas y aplicaciones prototipo para medir escalabilidad y desempeño
- Establecer criterios para la selección de casos de uso.
- Metodología de desarrollo ágil: varios ciclos de análisis-diseño- implementación-pruebas.

5.2 Organización del proyecto

El proyecto se organiza en paquetes de trabajo y actividades dentro de cada paquete de trabajo. Cada paquete de trabajo produce un conjunto de entregables.

5.2.1 PT1: PaaS Hadoop

- Actividad 1.1: Configurar Hadoop en multinodo sobre máquinas virtuales en Vbox.
- Actividad 1.2: Reproducir configuración sobre SC3.
- Actividad 1.3: Definir aplicaciones prototipo y métricas de escalabilidad.
- Actividad 1.4: Medir escalabilidad y desempeño.
- Entregable 1.1: Máquina virtual VBox Hadoop multinodo.
- Entregable 1.2: Imagen de despliegue SC3 Hadoop multinodo.14
- Entregable 1.3: How to de instalación.
- Entregable 1.4: Reporte desempeño y escalabilidad –> referirse al entregable 3.2.
- Entregable 1.5 (opcional): Entregables 1.1 y 1.2 mejorados.

5.2.2 PT2: PaaS Spark

- Actividad 2.1: Configurar Spark en multinodo sobre máquinas virtuales en Vbox
- Actividad 2.2: Reproducir configuración sobre SC3
- Actividad 2.3: Definir aplicaciones prototipo y métricas de escalabilidad
- Actividad 2.4: Medir escalabilidad y desempeño
- Entregable 2.1: Máquina virtual VBox Spark multinodo
- Entregable 2.2: Imagen de despliegue SC3 Spark multinodo
- Entregable 2.3: Howto de instalación
- Entregable 2.4: Reporte desempeño y escalabilidad
- Entregable 2.5 (opcional): Entregables 2.1 y 2.2 mejorados

5.2.3 PT3: Casos de uso

- Actividad 3.1: Definición caso de uso para análisis de datos (Hadoop)
- Actividad 3.2: Ejecución y reporte caso de uso para análisis de datos.
- Actividad 3.3: Definición caso de uso para análisis predictivo (Spark).
- Actividad 3.4: Ejecución y reporte caso de uso para análisis predictivo.
- Entregable 3.1: Caso de uso para análisis de datos (Hadoop).
- Entregable 3.2: Ejecución y reporte caso de uso para análisis de datos.
- Entregable 3.3: Definición caso de uso para análisis predictivo (Spark).
- Entregable 3.4: Caso de uso para análisis predictivo.

5.3 Retos

- Hadoop es difícil de configurar y, además, existen pocos PaaS Hadoop y Spark reutilizables (p.ej. Cloudera).
- Ambas plataformas están diseñadas para correr bajo un modelo en el cual el cómputo va a los datos y, por tanto, bajo infraestructuras de cómputo en las cuales todos los nodos realizan funciones de cómputo y almacenamiento. Esto supone un punto de partida fundamentalmente distinto de computación clúster tradicional donde los recursos de cómputo están regularmente separados de los de almacenamiento.
- Reducir el tiempo que los investigadores tienen que gastar en configurar sus clústers Hadoop/Spark dada su compleja adecuación para correr sobre computación clúster.

6 IMPLEMENTACIÓN

6.1 Entorno Hardware

6.1.1 Maquinas virtuales sobre Vbox

Disponiendo de una maquina regular como se muestra en el Cuadro 2, se disponen dos maquinas virtuales en Vbox con la siguiente configuración:

	Arquitectura	RAM	Disco Duro	Cores/Nodo
Maquinas Vbox	3.20GHz Intel® Pentium(R) 4	512 Mb	8 Gb	2

Cuadro 1: Características de Hardware Vbox

6.1.2 Clúster de pruebas

El hardware utilizado fue proporcionado por la Universidad Industrial de Santander junto al centro de Supercomputación y Calculo Científico SC3. El clúster de prueba contó con cinco maquinas con las mismas características de hardware y una maquina adicional como se muestra en el Cuadro 2

	Arquitectura	RAM	Disco Duro	Cores/Nodo
Maquinas Regulares	3.20GHz Intel® Pentium(R) 4	2 Gb	160 Gb	2
Maquina Adicional	3.20GHz Intel® Pentium(R) 4	1 Gb	160 Gb	2

Cuadro 2: Características de Hardware Maquinas Fisicas

6.1.3 Supercomputador Guane

Arquitectura	RAM	Tarjeta de Video	Cores/Nodo	GPUs/Nodo
2,67 GHz Intel Xeon CPU E5640	104 Gb	Nvidia Tesla M2050	8	3584
2,40 GHz Intel Xeon CPU E5645	104 Gb	Nvidia Tesla M2050	12	3584

Cuadro 3: Caracteristicas de Hardware Guane

6.2 Entorno Software

6.2.1 Sistema Operativo

Como sistema base se utilizo CentOS 6.5, la ultima versión estable de este sistema CentOS es una distribución Linux que intenta proporcionar una plataforma estable, tipo empresarial, gratuito y mantenido por la comunidad que logra una compatibilidad binaria del 100% con Red Hat Enterprise Linux (RHEL) del cual CentOS es una bifurcación. Para 2014 el proyecto CentOS pasa a ser patrocinado por Red Hat.

6.2.2 Apache Ambari

Es un framework completamente abierto y operacional para la prestación, administración y monitoreo de clústers Hadoop. Incluye una colección intuitiva de herramientas de operador y un conjunto de APIs que enmascaran la complejidad de Hadoop, simplificando las operaciones del clúster. Además integra Spark.

6.3 Análisis de Implementación

Al poner en marcha el proyecto se efectúan varios cambios en la ejecución de las actividades mencionadas anteriormente, como se presenta a continuación:

6.3.1 Disposición de maquinas virtuales en Vbox

Teniendo en cuenta el análisis previo a la implementación del proyecto y el análisis de las pruebas que se realizarían en un futuro, se toma la decisión de no utilizar maquinas virtuales con recursos computacionales tan limitados (hardware virtual), ya que el numero de maquinas virtuales montadas en una sola maquina regular, Cuadro2 se vería ciertamente limitado y el desarrollo del prototipo en sus etapas iniciales no hubiese sido el adecuado. Debido a estos planteamientos y dada la aprobación del profesor raúl ramos pollán, director de proyecto, se procedió a realizar todo el proceso de prototipado en maquinas físicas.

6.3.2 Utilización de Apache Ambari

Otra de las cosas a tener en cuenta fue la utilización de Apache Ambari como recurso esencial para la prestación, administración y monitoreo de clústers Hadoop, ya que integra tres herramientas esenciales e importantes para el desarrollo del proyecto: Hadoop, MapReduse y Spark, un framework completamente abierto y sin ninguna limitación a la hora de ofrecer sus servicios para Big Data. Herramientas como Cloudera tienen ciertas limitaciones a la hora de ofrecer sus servicios, por lo que se opta por utilizar Apache Ambari.

6.3.3 Reorganización de Actividades

Gracias a las posibilidades de integrar Hadoop y Spark en un paquete de despliegue como lo permite Apache Ambari además de su implementación en maquinas fisicas se realizo una reorganización de actividades, dejando un paquete de trabajo único con cinco actividades por implementar y un conjunto de cinco entregables:

PT1: PaaS Hadoop-Spark

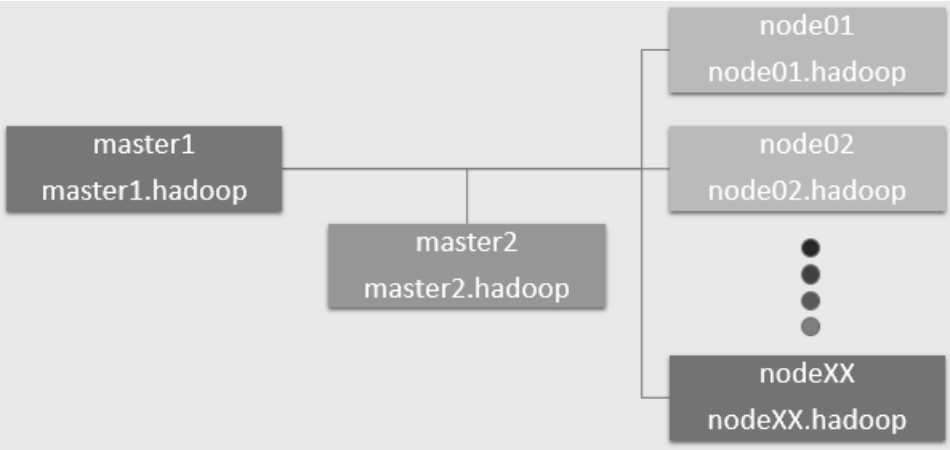
- Actividad 1.1: Configurar Hadoop y Spark en multinodo sobre máquinas físicas con repositorios en la web.
- Actividad 1.2: Configurar Hadoop y Spark en multinodo sobre máquinas físicas con repositorios locales.
- Actividad 1.3: Reproducir configuración sobre SC3.
- Actividad 1.4: Definir aplicaciones prototipo y métricas de escalabilidad.
- Actividad 1.5: Medir escalabilidad y desempeño.
- Entregable 1.1: Imagen de despliegue sobre SC3 Hadoop-Spark multinodo, master.
- Entregable 1.2: Imagen de despliegue sobre SC3 Hadoop-Spark multinodo, worker.
- Entregable 1.3: How to de instalación Hadoop y Spark en multinodo sobre máquinas físicas con repositorios en la web.
- Entregable 1.4: How to de instalación Hadoop y Spark en multinodo sobre máquinas físicas con repositorios locales.
- Entregable 1.5: How to despliegue de imágenes master-worker sobre guane.
- Entregable 1.6: Reporte desempeño y escalabilidad sobre despliegue en Guane.

6.4 Desarrollo de Actividades

6.4.1 Configuración equipos nativos

Se utilizaron maquinas físicas como se muestra en el Cuadro 2 dispuestas de la siguiente manera:

Figura 12: Disposición Clúster Hadoop de Prueba



Fuente: Autores.

5 maquinas regulares y 1 maquina adicional para un total de 6 maquinas las cuales conforman nuestro cluster de pruebas.

6.4.2 Configuración sistema operativo CentOS 6.5 sin entorno gráfico

Se instalo CentOS 6.5 sin entorno gráfico para minimizar el uso de recursos computacionales sobre las maquinas dispuestas para el clúster, ya que solo requiere un uso en memoria de 392 M CLI [21]. Dos particiones primarias, la primera partición para swap con un total de 4 Gb y la segunda con el espacio restante de disco para el sistema de archivos. los hostnames de las maquinas quedaron de la siguiente manera:

hosts master: master1 y master 2

host esclavos: node01, node02, node03, node04

por comodidad se asigno una misma contraseña de usuario root para todas las maquinas.

6.4.3 Configuración e instalación de Software en el Clúster de prueba

Una vez terminada la instalación del sistema operativo se procede a la configuración e instalación del software necesario para que Apache Ambari pueda funcionar correctamente.

A continuación se muestra los comandos que se ejecutaron en cada una de las maquinas pertenecientes al clúster:

- **Activar puerto de red:**

#ifup eth0

Asumiendo que eth0 es el puerto predeterminado

- **Instalar programas necesarios:**

#yum install nano wget openssl ntp

Se utilizo nano para la edición de archivos.

- **Editar el archivo ifcfg-eth0:**

#nano /etc/sysconfig/network-scripts/ifcfg-eth0

Y se cambio la linea “ONBOOT=no” por “ONBOOT=yes”

- **Activar servicio SSH y NTP:**

#chkconfig sshd on

#chkconfig ntpd on

#ntpdate 0.rhel.pool.ntp.org

- **Desactivar iptables:**

#chkconfig iptables off

#/etc/int.d/iptables stop

- **Editar el archivo hosts:**

#nano /etc/hosts

Nuestro archivo “hosts” se edito de la siguiente manera, con dos master y dos nodos esclavos:

192.168.66.187 master1.hadoop master1

192.168.66.194 master2.hadoop master2

192.168.66.180 node01.hadoop node01

192.168.66.176 node02.hadoop node02

192.168.66.176 node02.hadoop node03

192.168.66.176 node02.hadoop node04

Y así sucesivamente con todos los host que se quieran incluir en el clúster. Fue necesario utilizar un FQDN (nombre de dominio completo) apropiado, el instalador de Apache Ambari no reconoce las IPs asignadas a cada host, reconoce los nombres de dominio completos (node01.hadoop).

- **Enviar fichero /etc/host a todas las maquinas del cluster:**

#scp /etc/hosts root@<hostname>:/etc/hosts

Para evitar la edición de cada fichero en cada maquina y ahorrar tiempo se envio el archivo “host” via SCP.

- **Desactivar SELINUX:**

#nano /etc/selinux/config

Y se cambio la línea “SELINUX=enforcing” por “SELINUX=disabled”. Para evitar la edición de cada fichero en cada maquina y ahorrar tiempo se envió el archivo “config” vía SCP.

#scp /etc/selinux/config root@<hostname>:/etc/selinux/config

- **Reiniciar todos los hosts:**

#reboot

- **Generar clave SSH:**

#ssh-keygen

Y se envió la clave SSH publica a todos los host del clúster incluyendo al propio master1.

#ssh-copy-id -i ~/.ssh/id_rsa.pub root@<hostname>

password:XXXXXX

6.4.4 Pre-instalacion y Pre-configuracion de Apache Ambari

Una vez realizada la configuración en instalación del software necesario para generar un ambiente favorable para Apache Ambari se procedió a pre-configurar el servidor Ambari:

- **Instalar servidor Ambari:**

Se descargo el siguiente repositorio:

#cd /etc/yum.repos.d/

#wget http://public-repo-1.hortonworks.com/ambari/centos6/1.x/updates/1.5.1/ambari.repo

Se instalo el servidor ambari:

#yum install ambari-server

Se configuro el servidor Ambari en modo silencioso o predeterminado:

#ambari-server setup –s

Se inicio ambari server:

#ambari-server start

Y por ultimo se reinico el master1:

#reboot

6.4.5 Instalación con Apache Ambari-repositorios en la web

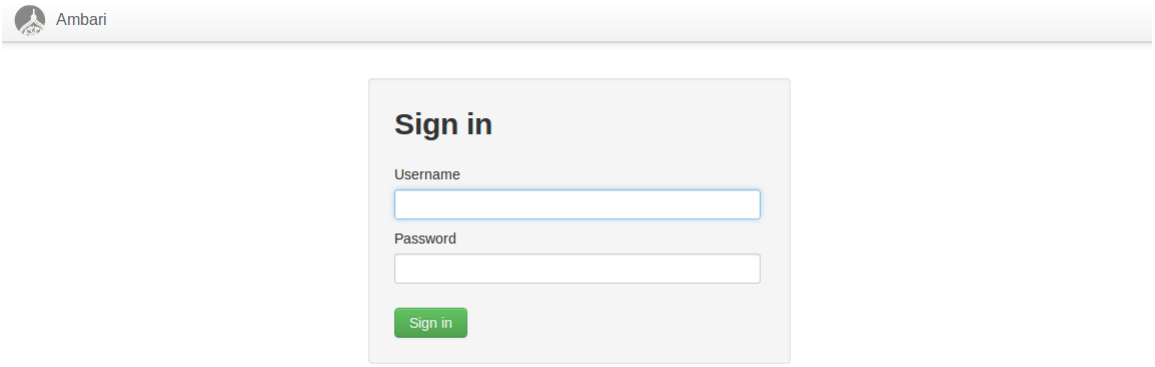
Se realizo la instalación final de Apache Ambari por medio de la aplicación web de Ambari (cluster install wizard) que funciona a través del puerto 8080, desde un navegador web se introdujo la IP correspondiente al master1 o su defecto el hostname:

http://192.168.66.187:8080

http://master1.hadoop:8080

Se cargo la siguiente pagina de Inicio-login, Figura 13:

Figura 13: Pagina Inicio-Login



Fuente: Autores.

Ambari trae un usuario predeterminado al igual que una contraseña predeterminada:

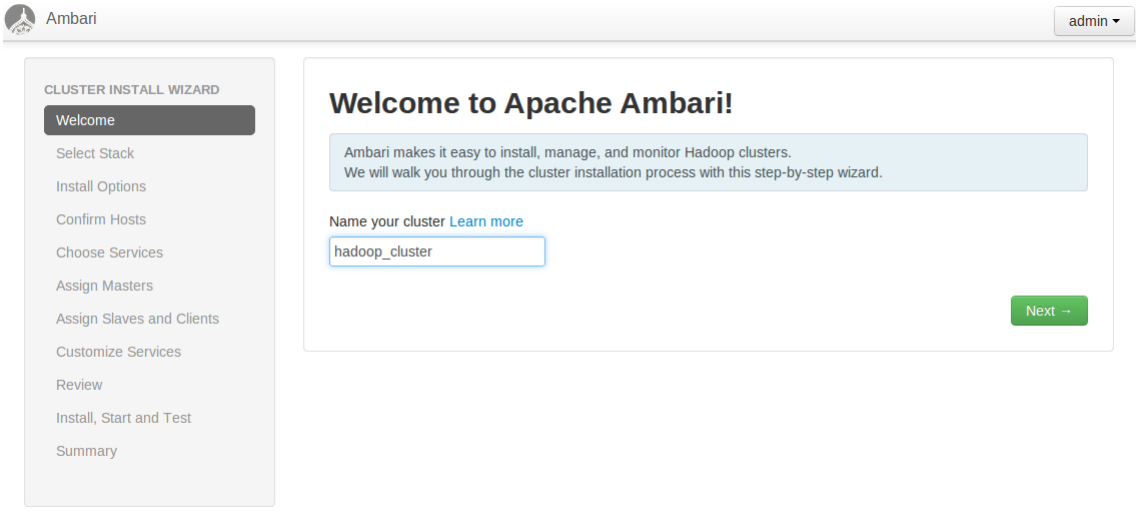
Username: admin

Password: admin

A continuación se muestra el asistente de instalación de clúster (cluster install wizard), Figura 14:

Como primer paso nombramos nuestro clúster, en nuestro caso lo nombramos “hadoop_cluster”.

Figura 14: Nombre de Clúster

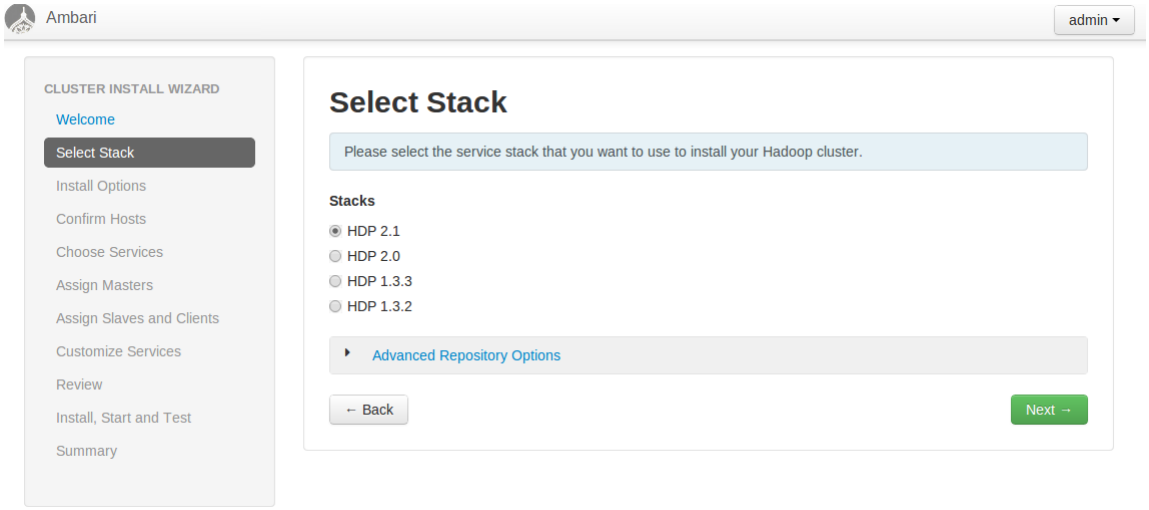


Fuente: Autores.

Luego seleccionamos la plataforma de datos mas actualizada de Apache Ambari, Figura 15:

Se selecciono la ultima versión de la plataforma de datos, HDP 2.1, correspondientes a los repositorios ubicados en la web.

Figura 15: Plataforma de Datos



Fuente: Autores.

Luego incluimos los hosts pertenecientes a nuestro clúster usando los FQDNs (nombres de dominio completo), Figura 16, además de la llave SSH privada generada anteriormente

(sección 7.3.2.2-Generar Clave SSH), Figura 17. La llave privada se encuentra en el home del master, carpeta .ssh, fichero id_rsa, se copio la clave privada y la se pego directamente recuadro correspondiente, Figura 18:

Para nuestro Clúster tuvimos las siguiente disposición:

master1.hadoop

master2.hadoop

node01.hadoop

node02.hadoop

node03.hadoop

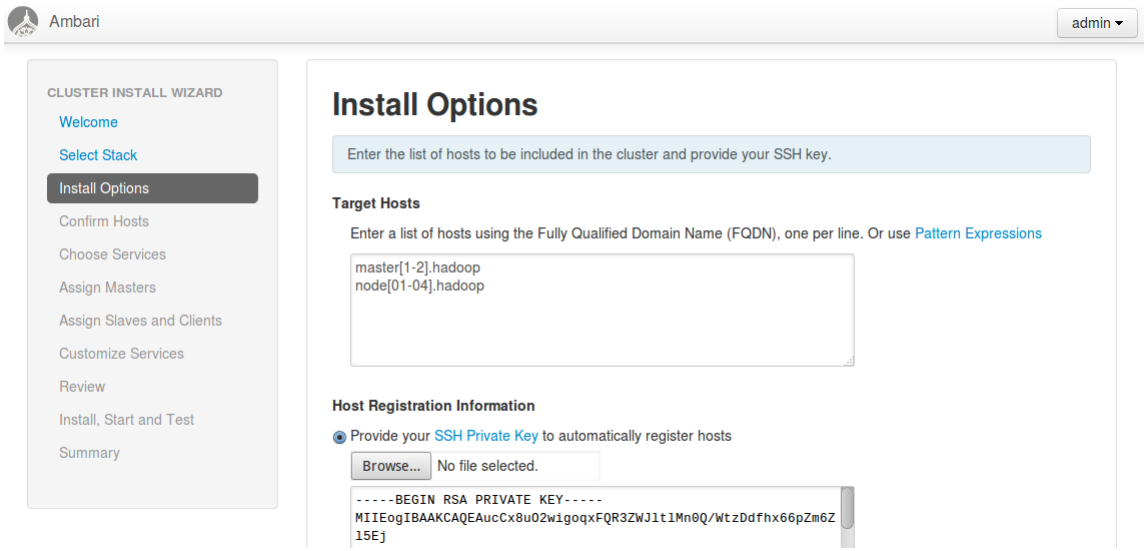
node04.hadoop

Nombres de dominio completo los cuales se abreviaron de la siguiente manera para evitar edición individual ya que en caso de manejar clústers con mas de 10 hosts correspondería a una tarea tediosa:

master[1-2].hadoop

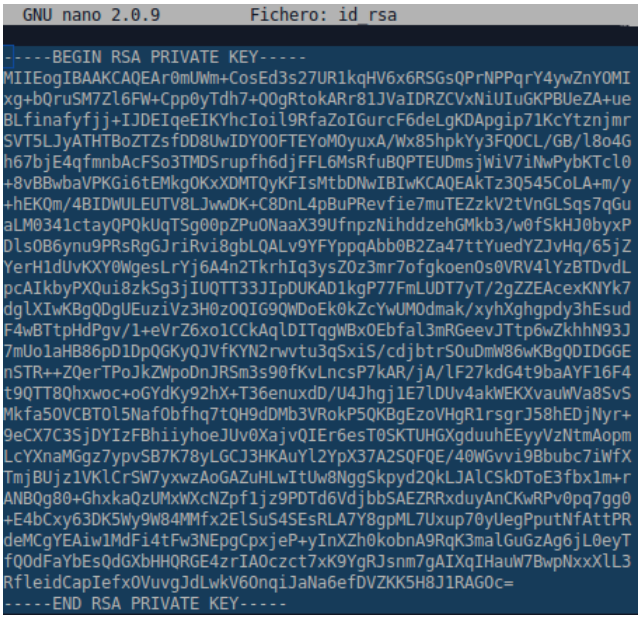
node[01-04].hadoop

Figura 16: Inclusion de Hosts



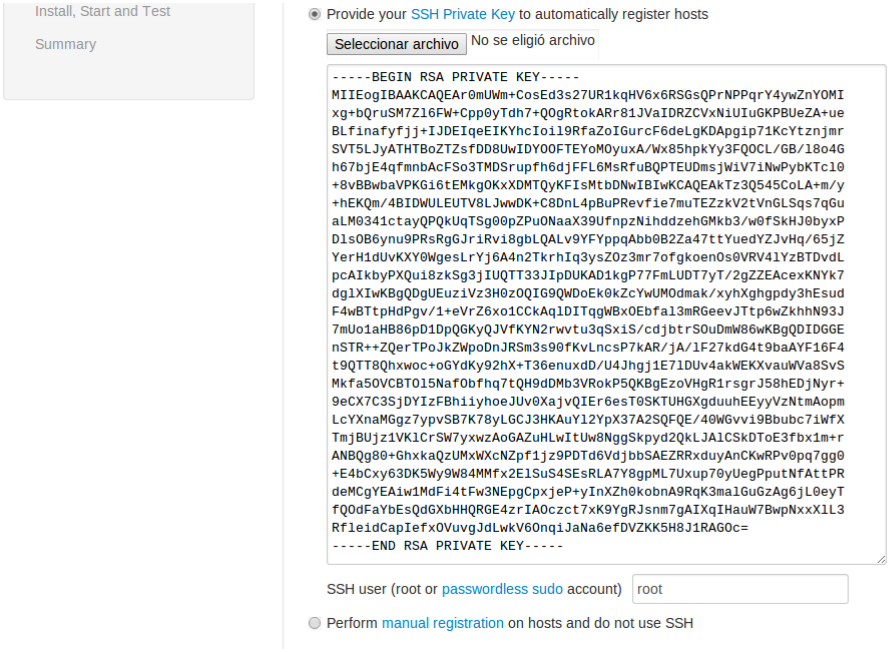
Fuente: Autores.

Figura 17: Llave Privada SSH



Fuente: Autores.

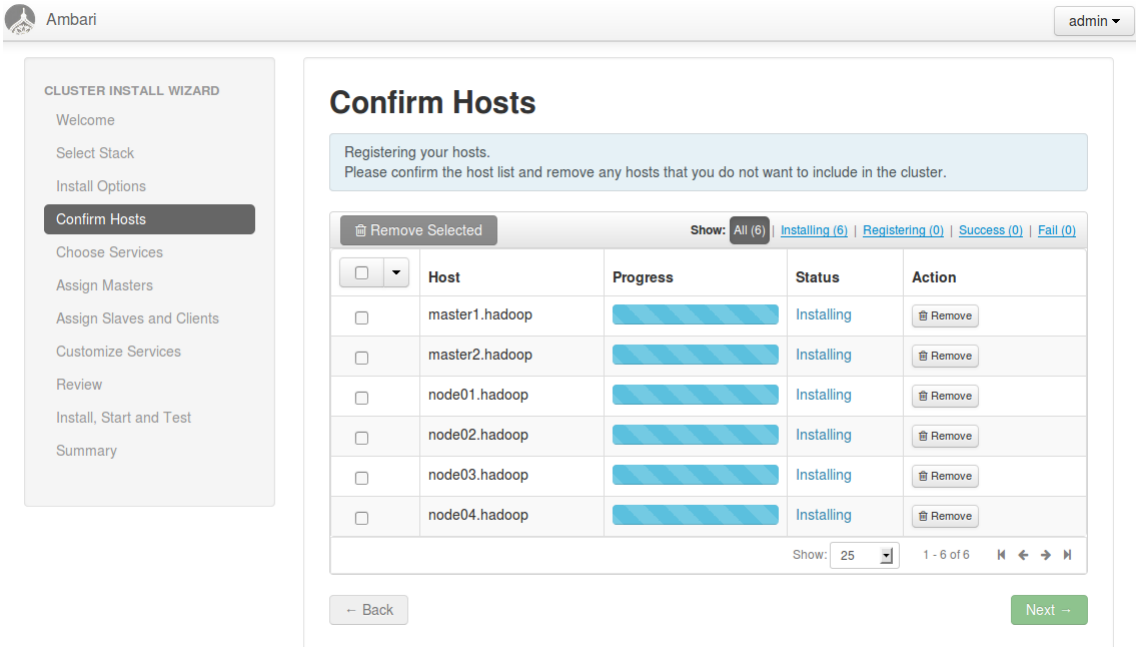
Figura 18: Inclusión de Llave Privada



Fuente: Autores.

A continuación se realizo el registro de los hosts, Figura 19:

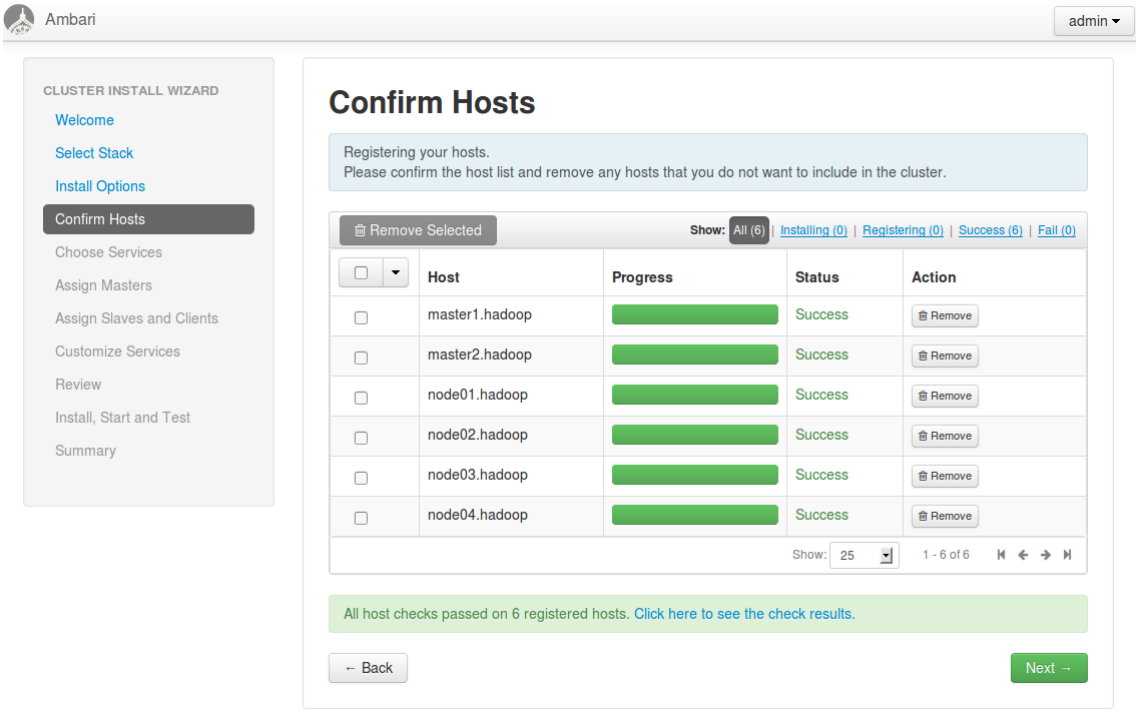
Figura 19: Registro de Hosts



Fuente: Autores.

Una ves realizado el registro con éxito se observo la siguiente confirmación, Figura 20:

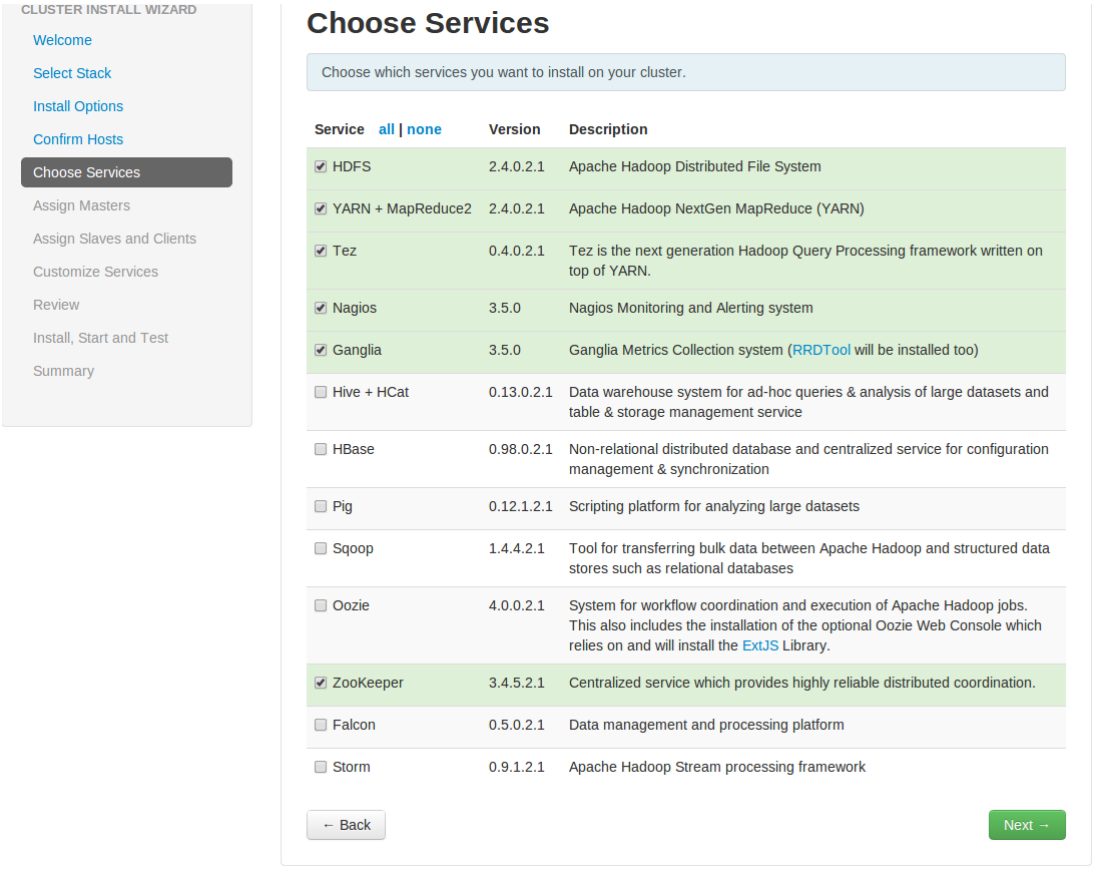
Figura 20: Confirmación de Registro



Fuente: Autores.

Luego de realizar el registro correcto de nuestros hosts, se procede a realizar la selección de servicios que se instalaron en nuestro clúster de prueba, Figura 21, se realizo una instalación de los servicios acordados como objetivos para el proyecto como el sistema de archivos de Haddop HDFS y Hadoop-MapReduce, agregando algunos otros como servicios de monitoreo y alerta para el clúster:

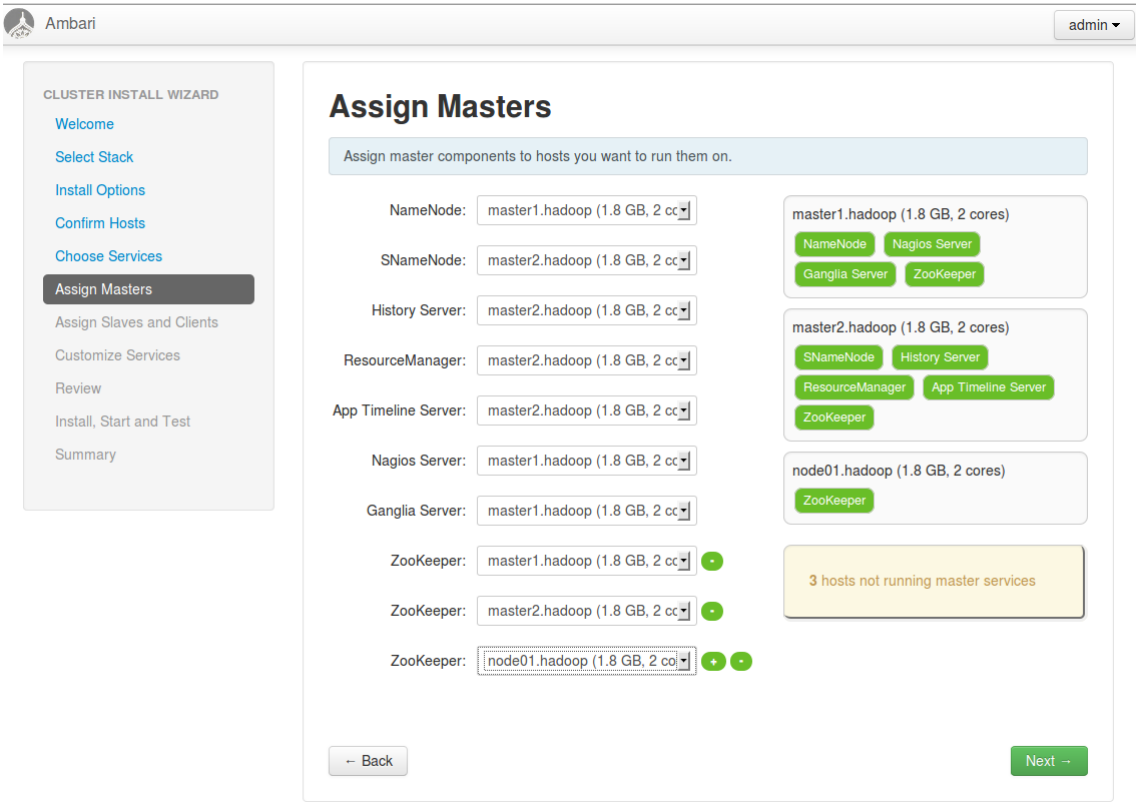
Figura 21: Selección de Servicios



Fuente: Autores.

A continuación se distribuyeron los servicios seleccionados, se realizo la siguiente distribución, Figura 22:

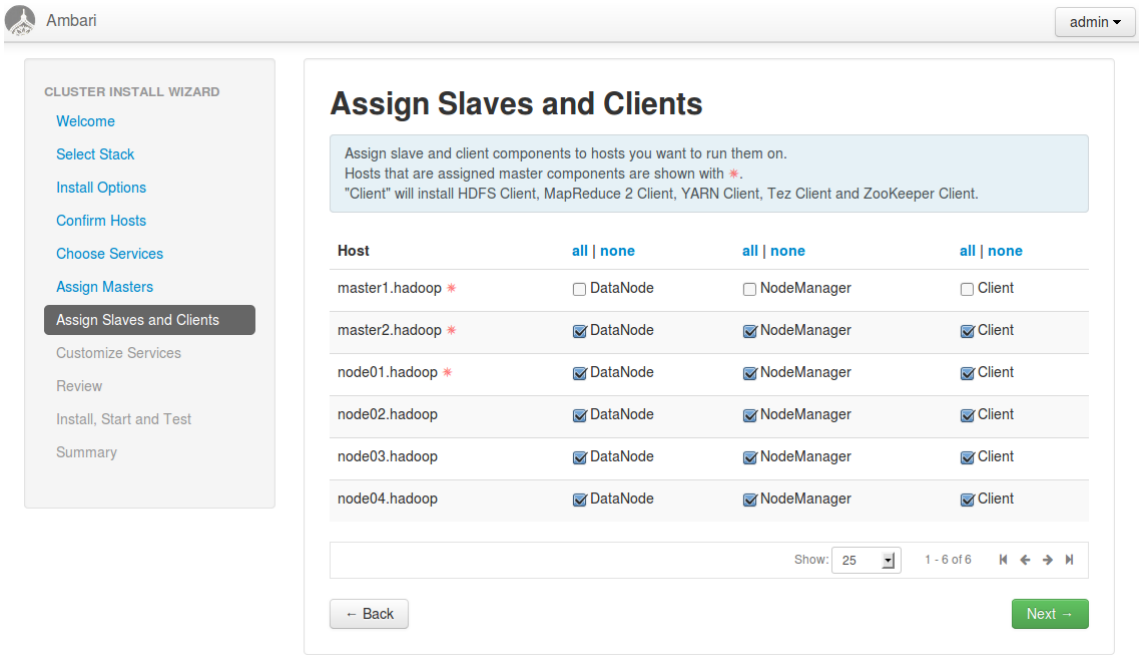
Figura 22: Distribución de Servicios



Fuente: Autores.

A continuación se realizo la selección de nodos esclavos, Figura 23:

Figura 23: Asignación de Componentes Cliente y Esclavo

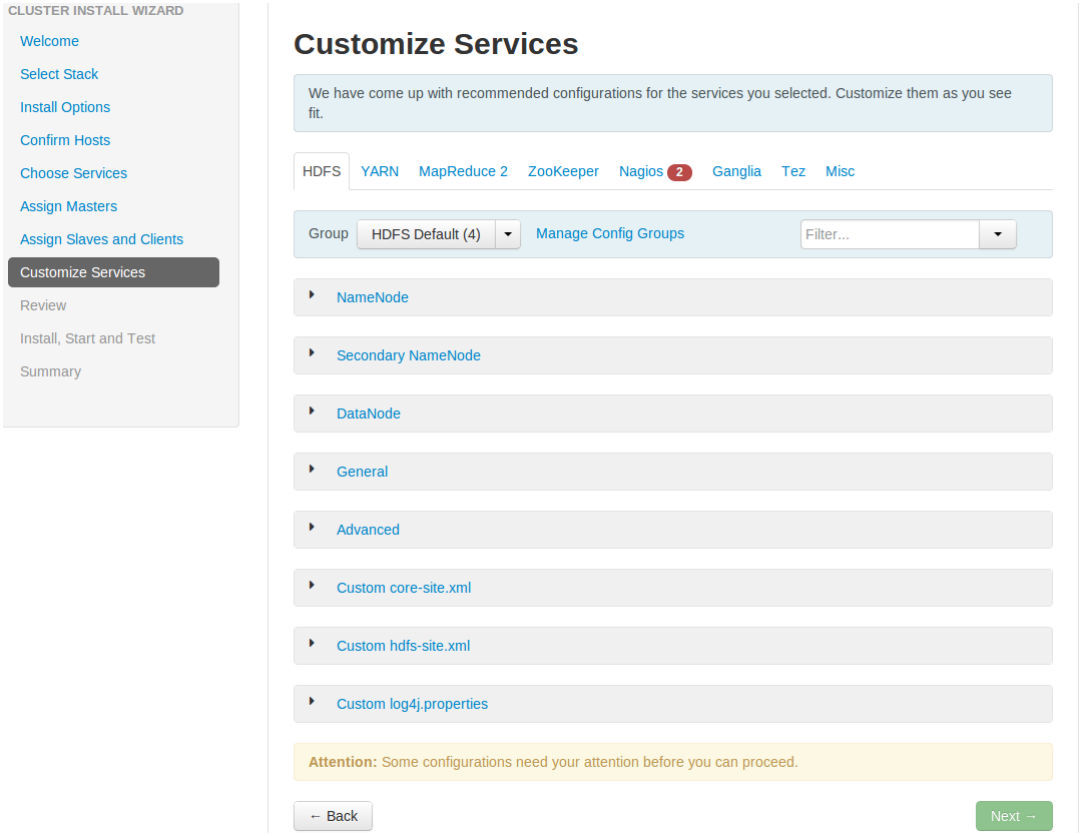


Fuente: Autores.

Se seleccionaron los hosts que no contienen servicios de master y que contendrán los servicios de cliente y ejecutaran las tareas o jobs.

Como siguiente paso tuvimos la personalización de cada uno de los servicios a instalar en el clúster, Figura 24:

Figura 24: Personalización de servicios

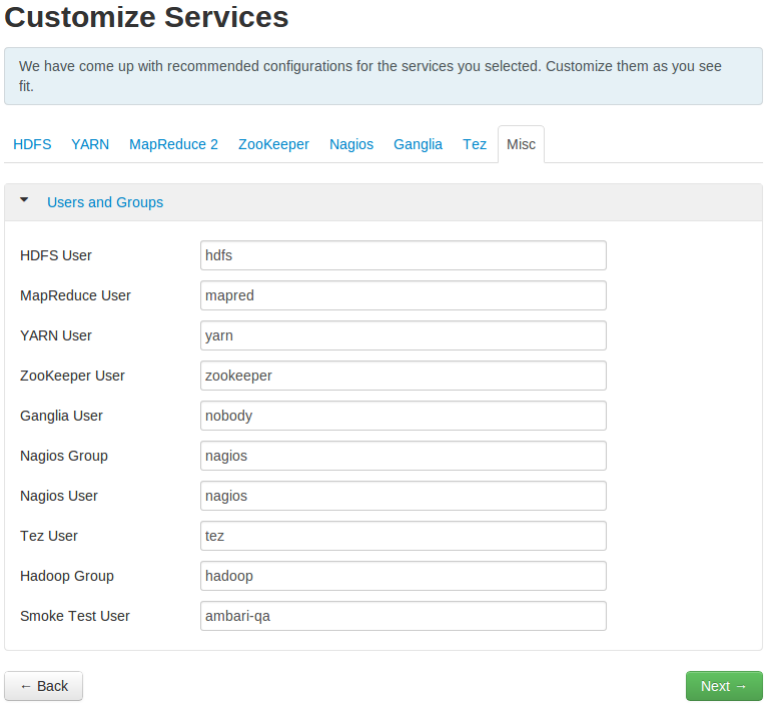


Fuente: Autores.

En el proceso de instalación se utilizo la configuración por defecto de cada uno de los servicios disponibles, esto para evitar problemas futuros en el funcionamiento del clúster.

En la pestaña Misc se muestran los usuarios y grupos de usuarios que se crean para cada servicio a instalar en el clúster, Figura 25:

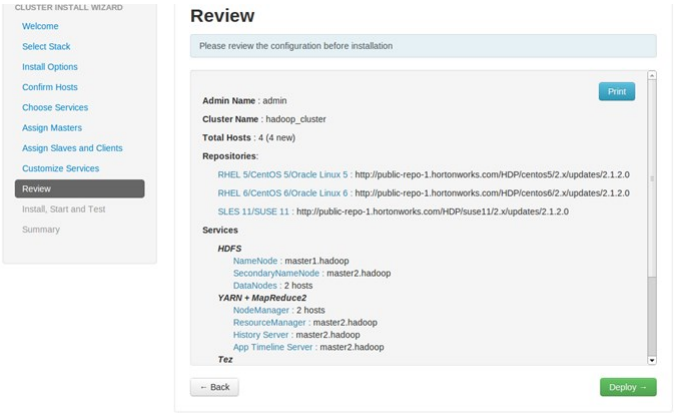
Figura 25: Usuarios y Grupos de Usuarios



Fuente: Autores.

A continuación se genero un archivo de revisión en el cual se describe de una manera general la instalación a ejecutar, como también la ubicación de cada uno de los servicios en el clúster, Figura 26:

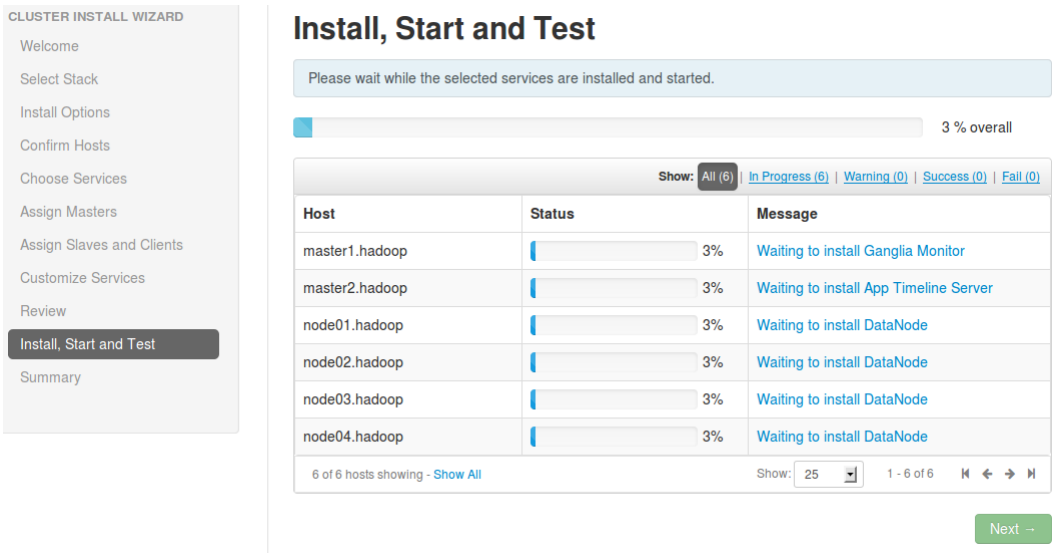
Figura 26: Revisión de Instalación



Fuente: Autores.

Por ultimo se dio comienzo a la instalación de los servicios configurados, Figura 27:

Figura 27: Instalación Final



Fuente: Autores.

Una ves terminada el proceso de instalación en el clúster de pruebas se da inicio automático a todos los servicios configurados. Con este ultimo proceso termino la instalación de Apache Ambari, integrando el sistema de archivos HDFS y Hadoop MapReduce, como se había previsto en un principio.

6.4.6 Instalación de Spark

- **Descargar Spark**

Directamente sobre el master se descargo el archivo correspondiente a Spark:

#wget http://d3kbcqa49mib13.cloudfront.net/spark-1.1.0-bin-hadoop2.4.tgz

- **Descomprimir archivo**

Se descomprimió el archivo descargado, se renombro la carpeta extraída y se movió a la raíz del sistema :

#unzip spark-1.1.0-bin-hadoop2.4.tgz -d ~/

#mv spark-1.1.0-bin-hadoop2.4.tgz spark

#mv spark /spark

- **Copiar variables de entorno**

cp ~/.bashrc /home/hdfs

- **Agregar workers con hostname**

nano /spark/conf/slaves

- **Eviar archivo slave a los workers**

scp /spark/conf/slaves <workers>:/spark/conf/slaves

- **Modificar la plantilla spark-env.sh.template**

cp /spark/conf/spark-env.sh.template /spark/conf/spark-env.sh

nano /spark/conf/spark-env.sh

Se agrego la siguiente linea:

export SPARK_EXECUTOR_INSTANCES=<num-workers>

Y se cambio <num-workers> por la cantidad de workers que se usaron.

- **Enviar a workers**

scp /spark/conf/spark-env.sh <workers>:/spark/conf/spark-env.sh

cd /spark

./bin/spark-shell –master yarn-client

En este punto Spark esta instalado, configurado y listo para usar.

6.4.7 Instalación con Apache Ambari-repositorios locales

Al tener que realizar una instalación con repositorios disponibles en la web se crea una dependencia en términos de tiempo de descarga a la hora de realizar dicha instalación, dependiendo ampliamente de la conexión a internet que se posea; para evitar esta dependencia se puede crear una carpeta que contenga estos repositorios localmente, y de esta manera reducir notablemente los tiempos de instalación.

Para esto se realizo de nuevo los pasos ilustrados en las secciones 6.4.1, 6.4.2 y 6.4.3, a partir de esta configuración se realizaron los siguientes pasos para crear los repositorios locales, y en base a esto realizar la instalación de Apache Ambari con los repositorios alojados localmente en el master1.

- **Instalar Repo de Ambari y HDP**

Se instalaron los .repo de ambari y HDP en la siguiente carpeta:

#cd /etc/yum.repos.d/

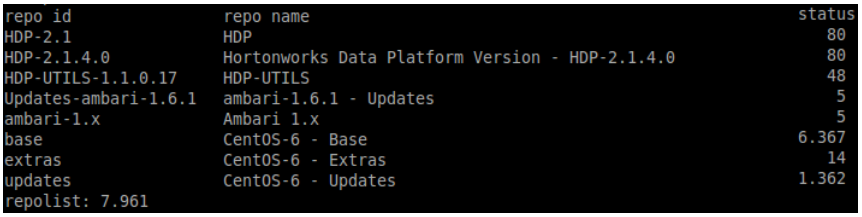
#wget http://s3.amazonaws.com/public- repo-1.hortonworks.com/HDP/ centos6/2.x/2.1-latest/hdp.repo

#wget http://public-repo-1.hortonworks.com/ ambari/centos6/1.x/updates/1.6.1/ ambari.repo

y se verifico la disposición del los repositorios:

#yum repolist

Figura 28: Lista de Repositorios



Fuente: Autores.

- **Crear repositorios locales de Ambari**

De los repositorios que se muestran en la Figura 28, nos interesaron, Updates-ambari-1.6.1 y ambari-1.x.

Creamos un directorio en el cual se almacenaron los paquetes de los repos:

#mkdir /repos

Se utilizo reposync para traer los paquetes de los repos al directorio creado anteriormente:

#reposync –r Updates-ambari-1.6.1 –p /repos

#reposync –r ambari-1.x –p /repos

Una vez que los paquetes fueron traídos al directorio fue necesario crear el metadata del repo para ser reconocido por yum:

#createrepo /repos/Updates-ambari-1.6.1/

#createrepo /repos/ambari-1.z/

Luego se modifico el siguiente archivo:

#nano /etc/yum.repos.d/ambari.repo

Se cambiaron las URLs remotas asignadas a los repositorios en la web por las direcciones del servidor local el cual es el master1. En el archivo original se tenia la siguiente URL:

baseurl=http://public-repo-1.hortonworks.com/ambari/centos6/1.x/GA

Y se cambio por la URL asignada al repositorio local:

baseurl=http://master1/repos/ambari-1.x/

Y por ultimo se creo un enlace simbólico entre la carpeta donde se encuentran los repositorios y la carpeta que alberga los archivos del servidor local:

#cd /var/www/html

#ln –s /var/www/html/ /repos

En este punto los repositorios ya estaban disponibles de manera local en el master1, listos para ser usados en el proceso de instalación de Apache Amabari. Ademas la dependencia en términos de tiempo de descarga a la hora de realizar la instalación con los repositorios ubicados en la web fue eliminada.

- **Descargar JDK para instalar Java localmente**

Luego se descargo jdk-7u45-linux-x64.tar.gz y se guardo el paquete en /var/lib/ambari-server/resources para evitar que Ambari busque el paquete ubicado en la web y lo consiga localmente.

#wget http://public-repo-1.hortonworks.com/ ARTIFACTS/jdk-7u45-linux-x64.tar.gz

Luego se aplico el siguiente comando:

#ambari-server setup –s

Y automáticamente se instala jdk en la carpeta /usr/jdk64

Luego se procedio a instalar JAVA utilizando Alternatives:

#alternatives —install /usr/bin/jar jar /usr/jdk64/jdk1.7.0_45/bin/jar 2

#alternatives —install /usr/bin/javac javac /usr/jdk64/jdk1.7.0_45/bin/javac 2

#alternatives —set javac /usr/jdk64/jdk1.7.0_45/bin/jar

#alternatives —set javac /usr/jdk64/jdk1.7.0_45/bin/javac

y se comprobó la versión actual de JAVA:

#java –versión

Por ultimo se aplicaron las variables de entorno de instalación:

#export JAVA_HOME=/usr/jdk64/jdk1.7.0_45

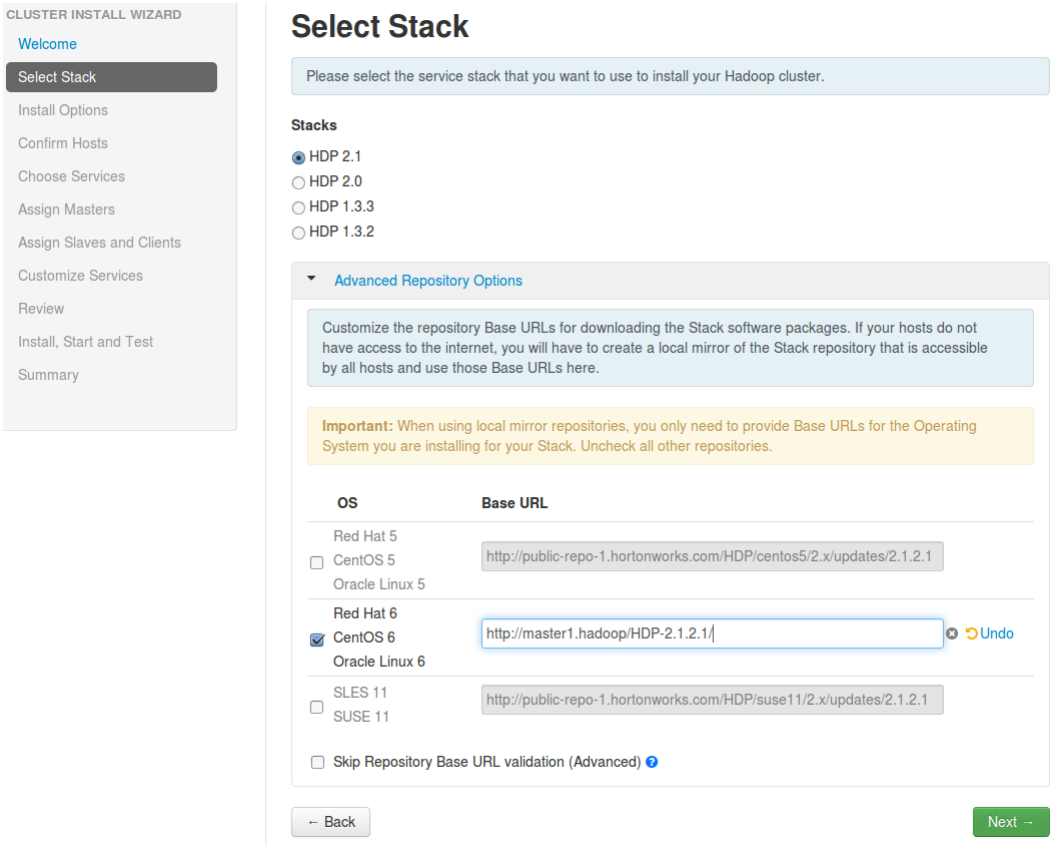
#export JRE_HOME=/usr/jdk64/jdk1.7.0_45/jre

#export PATH=$PATH:/usr/jdk65/jdk1.7.0_45/ bin:/usr/jdk64/jdk1.7.0_45/jre/bin

- **Instalación con Apache Ambari**

luego se retomo la instalación vista en la sección 6.4.4 (Pre-instalacion y Pre-configuracion de Apache Ambari) y luego se continuo con la sección 6.4.5 (Instalación Apache Ambari con repositorios en la web), solo que al momento de seleccionar la plataforma de datos mas actualizada de Apache Ambari, correspondientes a los repositorios ubicados en la web, se cambio por la ubicación local de los repositorios creados localmente, como se muestra en la Figura 29:

Figura 29: Cambio de repositorios en la web por los repositorios locales



Fuente: Autores.

Y se realizo la instalación correspondiente con los repositorios ubicados localmente en el clúster de pruebas.

6.4.8 Instalación de Spark

Se realizo la misma instalación realizada en la sección 6.4.6

6.4.9 Reporte de Actividades Realizadas

Hasta este punto de la implementación del proyecto se dio por terminada la curva de aprendizaje en relación a la configuración e instalación de Hadoop y Spark sobre un clúster de prueba con la utilización de Apache Ambari como herramienta de instalación:

- Actividad 1.1: Configurar Hadoop y Spark en multinodo sobre máquinas físicas con repositorios en la web.
- Actividad 1.2: Configurar Hadoop y Spark en multinodo sobre máquinas físicas con repositorios locales.

De esta curva de aprendizaje se anexan los siguientes entregables:

- Anexo 1: Entregable 1.3: How to de instalación Hadoop y Spark en multinodo sobre máquinas físicas con repositorios en la web.
- Anexo 2: Entregable 1.4: How to de instalación Hadoop y Spark en multinodo sobre máquinas físicas con repositorios locales.

6.4.10 Extracción de imágenes de despliegue sobre SC3

Para la extracción de las imágenes de despliegue se utilizo una maquina regular como se muestra en el Cuadro 2 con CentOS-6.5 con entorno gráfico, ya que para trabajos futuros se espera que usuarios que se encuentran en redes externas a la red de la UIS tengan la posibilidad de ver el entorno gráfico de las maquinas que vayan a configurar sobre Guane, y de esta manera acceder a las interfaces gráficas web generadas por los Frameworks disponibles por Ambari.

6.4.10.1 Imagen de despligue sobre SC3 Hadoop-Spark multinodo, worker

6.4.10.1.1 Configuración sistema operativo CentOS 6.5 con entorno gráfico Se instalo CentOS 6.5 con entorno gráfico sobre la maquina regular. Dos particiones primarias, la primera partición para swap con un total de 4 Gb y la segunda con el espacio restante de disco para el sistema de archivos. El hostname de la maquina se asigno como “worker00”. Como contraseña de usuario root se asigno “griduis”.

6.4.10.1.2 Configuración e instalación de Software Una vez terminada la instalación del sistema operativo se procede a la configuración e instalación del software necesario, antes de extraer la imagen de despliegue, para que Apache Ambari pueda funcionar correctamente.

A continuación se muestra los comandos que se ejecutaron en la maquina regular:

- **Instalar programas necesarios:**

#yum install nano wget openssl ntp

Se utilizo nano para la edición de archivos.

- **Activar servicio SSH y NTP:**

#chkconfig sshd on

#chkconfig ntpd on

#ntpdate 0.rhel.pool.ntp.org

- **Desactivar iptables:**

#chkconfig iptables off

#/etc/int.d/iptables stop

- **Desactivar SELINUX:**

#nano /etc/selinux/config

Y se cambio la línea “SELINUX=enforcing” por “SELINUX=disabled”.

- **Reiniciar la maquina:**

#reboot

- **Editar el archivo fstab**

#nano /etc/fstab

Se cambiaron los identificadores universales únicos “UUID” [23] pertenecientes a cada una de las particiones del disco, Figura 30, por las etiquetas de disco correspondientes a las particiones en las maquinas de despliegue en Guane, Figura31:

Figura 30: Archivo fstab Original

```
#
# /etc/fstab
# Created by anaconda on Wed Sep 10 10:17:59 2014
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=e3fbbceb-3aac-429b-ae50-84732fdb21b1 /                       ext4    defaults        1 1
UUID=0c5cf049-9987-467c-a852-03665c6ffab5 swap                    swap    defaults        0 0
tmpfs                   /dev/shm                tmpfs   defaults        0 0
devpts                  /dev/pts                devpts  gid=5,mode=620  0 0
sysfs                   /sys                    sysfs   defaults        0 0
proc                    /proc                   proc    defaults        0 0
```

Fuente: Autores.

Figura 31: Archivo fstab Editado

```
#
# /etc/fstab
# Created by anaconda on Wed Sep 1Screenshot - 171014 - 18:26:120 10:17:59 2014
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/sda4               /                       ext4    defaults        1 1
/dev/sda1               swap                    swap    defaults        0 0
tmpfs                   /dev/shm                tmpfs   defaults        0 0
devpts                  /dev/pts                devpts  gid=5,mode=620  0 0
sysfs                   /sys                    sysfs   defaults        0 0
proc                    /proc                   proc    defaults        0 0
```

Fuente: Autores.

- **Eliminar el archivo “70-persistent_net.rules”**

#rm -rf etc/udev/rules.d/70-persistent_net.rules

- **Montar el sistema de archivos en la carpeta /mnt**

#mount -o bind / /mnt

En este punto se termino la instalación y configuración de la maquina regular. Se procede a extraer la imagen de despliegue desde Guane.

6.4.10.1.3 Extracción imagen de despliegue desde Guane Para la extracción de la imagen de despliegue sobre Guane se solicito la creación de una cuenta para realizar el acceso a la plataforma y a los recursos computacionales del SC3 [22].

Una vez sobre Guane se ingresaron los siguientes comandos en la consola:

- **Se creo la carpeta “Imagenes”**

#mkdir Imagenes

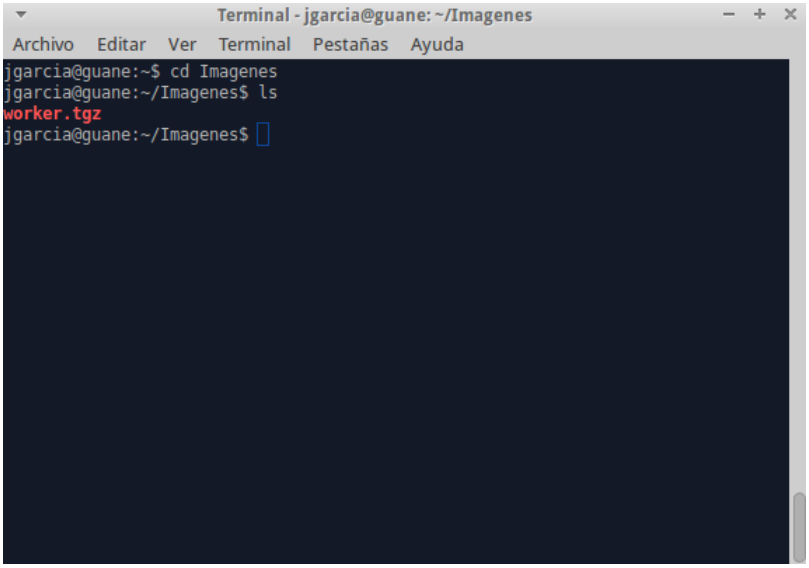
- **Extracción de imagen worker**

Se accedió a la carpeta Imagenes y se ingreso el siguiente comando para la extracción de la imagen de la maquina regular:

#ssh root@<hostname o ip maquina regular> ”cd /mnt; tar –posix –numeric-owner –one-file-system –exclude=’/mnt’ –exclude=’/proc/*’ –exclude=’/sys/*’ -czf - *” > worker.tgz

En este punto se finaliza la extracción de la imagen de despliegue worker sobre Guane, Figura 32.

Figura 32: Imagen worker Extraída Sobre Guane



Fuente: Autores.

6.4.10.2 Imagen de despliegue sobre SC3 Hadoop-Spark multinodo, master Para la imagen de despliegue sobre Guane, master, se utilizo la misma maquina regular en la cual se realizo la configuración e instalación de software, secciones 6.4.10.1.1 (en este caso el hostname se cambio a “master”) y 6.4.10.1.2. Agregado a este proceso se realizo la configuración y creación de los repositorios locales para Apache Ambari, sección 6.4.7 (los pasos: Instalar Repo de Ambari y HDP, Crear repositorios locales de Ambari y Descargar JDK para instalar Java localmente) y se descargaron los archivos correspondiente a Spark, sección 6.4.6 (solo los pasos: Descargar Spark y Descomprimir archivo) .

En este punto se termino la instalación y configuración de la maquina regular. Se procede a extraer la imagen de despliegue desde Guane.

6.4.10.2.1 Extracción imagen de despliegue desde Guane Para la extracción de la imagen de despliegue sobre Guane, se ingreso con la misma cuenta registrada anteriormente para realizar el acceso a la plataforma y a los recursos computacionales del SC3.

Una vez sobre Guane se ingresaron los siguientes comandos en la consola:

- **Se ingreso en la carpeta “Imagenes”**

#cd Imagenes

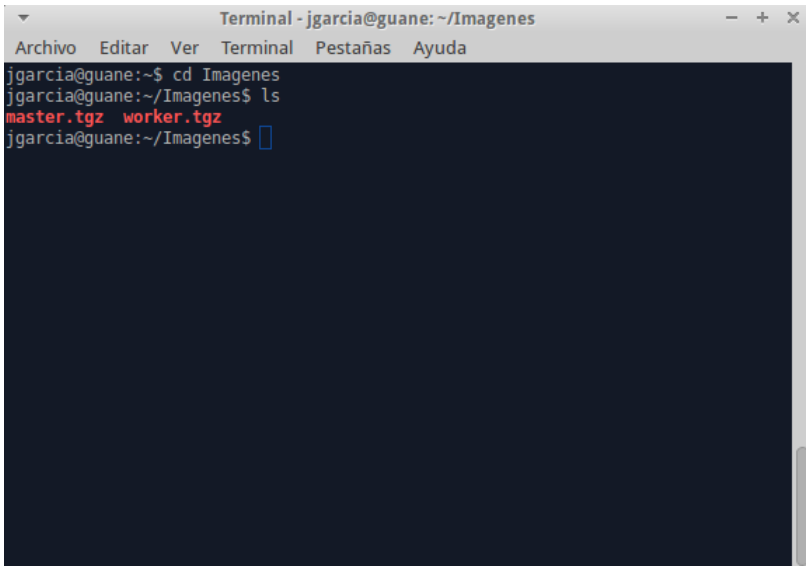
- **Extracción de imagen master**

se ingreso el siguiente comando para la extracción de la imagen de la maquina regular:

#ssh root@<hostname o ip maquina regular> ”cd /mnt; tar –posix –numeric-owner –one-file-system –exclude=’/mnt’ –exclude=’/proc/*’ –exclude=’/sys/*’ -czf - *” > master.tgz

En este punto se finaliza la extracción de la imagen de despliegue master sobre Guane, Figura 33.

Figura 33: Imagen master Extraída Sobre Guane



Fuente: Autores.

6.4.11 Reporte de Actividades Realizadas

Hasta este punto de la implementación del proyecto se dio por terminada la extracción de las imágenes de despliegue master-worker sobre SC3:

- Actividad 1.3: Reproducir configuración sobre SC3.

Además se cumplió con los siguientes entregables:

- Entregable 1.1: Imagen de despliegue sobre SC3 Hadoop-Spark multinodo, master.
- Entregable 1.2: Imagen de despliegue sobre SC3 Hadoop-Spark multinodo, worker.
- Anexo 3: Entregable 1.5: How to despliegue de imágenes Master-Worker sobre Guane.

7 PRUEBAS Y ANÁLISIS DE RESULTADOS

7.1 Definición de Escalabilidad

El software o hardware escalable es aquel que puede expandirse para soportar cargas de trabajo crecientes. Esta capacidad le permite al equipamiento de computo y software crecer con el tiempo, en lugar de ser reemplazados

El hardware escalable puede referirse a un único computador, una gran red de computadores u otro tipo de hardware. La escalabilidad de un sólo computador depende de que tan expansible sea. De acuerdo con esto, las palabras escalable, expansible y actualizable se pueden utilizar indistintamente. Por ejemplo, un computador puede tener múltiples bahías para discos duros, luego su capacidad de almacenamiento puede ser incrementada agregando dispositivos de almacenamiento. Un computador que incluya múltiples ranuras PCI tiene capacidad escalable de gráficos y E/S pues las tarjetas PCI pueden ser cambiadas o actualizadas. En cada caso el hardware escalable se puede expandir para suplir demandas crecientes

El software escalable típicamente hace referencia a aplicaciones empresariales que se pueden adaptar a una creciente cantidad de datos o a un numero creciente de usuarios. Por ejemplo, un sistema de administración de base de datos debería ser capaz de expandirse eficientemente a medida que mas datos son agregados. Un software escalable de Web hosting debería agregar nuevos usuarios y nuevas cuentas hosting con facilidad. Esto significa que el software escalable toma pocos espacio y recursos para pequeños usuarios, pero puede crecer eficientemente a medida que mas demanda es localizada en el software.

La escalabilidad de hardware y software en muy importante en términos monetarios. Es típicamente mas económico actualizar los sistemas actuales que reemplazarlos. Aunque todo el hardware y software tiene limitaciones, el equipamiento hardware y software escalable ofrece una ventaja a largo plazo sobre aquellos que no son diseñados para crecer en el tiempo.

7.2 Hadoop Benchmark Suit - HiBench

Se hicieron varias pruebas las cuales toman como métrica la toma del tiempo empleado en ejecutar determinadas tareas con el fin de medir la escalabilidad a medida que se agregan nodos de trabajo al clúster. Para estas pruebas se empleó un Benchmark Suit estándar para Hadoop que contiene varias cargas de trabajo ampliamente utilizados para el testeo de este tipo de clústers. A continuación se describe con mas detalle este Benchmark.

Esta suit contiene nueve diferentes programas Hadoop que evalúan el framework en términos de velocidad, rendimiento, ancho de banda de HDFS, utilización de recursos del sistema y patrones de acceso a datos.

7.2.1 Micro Benchmark

7.2.1.1 Sort Este trabajo ordena datos en forma de texto generado por un ejemplo de Hadoop incluido en la instalación.

7.2.1.2 WordCount Este trabajo cuenta la ocurrencia de cada palabra en un texto determinado. Este trabajo es representativo de otra clase típica de trabajos MapReduce del mundo real, los cuales extraen una pequeña cantidad de datos interesantes de un conjunto enorme de datos.

7.2.1.3 TeraSort TeraSort es una prueba estándar creada por Jim Gray. Las entradas para este test son generadas por TeraGen un programa de ejemplo incluido en la instalación.

7.2.2 HDFS Benchmarks

7.2.2.1 Enhanced DFSIO Enhanced DFSIO pone a prueba el rendimiento de HDFS generando un gran numero de tareas realizando escritura y lectura simultáneamente. Esta prueba mide la taza media de E/S, el rendimiento medio y el rendimiento agregado de cada tarea map en un clúster HDFS.

7.2.3 Web Search Benchmarks:

7.2.3.1 Nutch indexing La indexación de búsqueda a gran escala es uno de mas importantes usos de MapReduce. Este trabajo prueba el subsistema de indexación de Nutch, un popular motor de búsqueda de código abierto. Esta prueba utiliza datos web generados automáticamente cuyos hiperlinks y palabras siguen la distribución Zipfian con parámetros correspondientes. El diccionario utilizado para generar el texto de la pagina web es el archivo de diccionario por defecto de Linux /usr/share/dict/linux.words.

7.2.3.2 PageRank Las pruebas contienen una implemetación del algoritmo PageRank en Hadoop. Esta prueba utiliza datos web generados automáticamente cuyos hiperlinks y palabras siguen la distribución Zipf.

7.2.4 Machine Learning Benchmarks:

7.2.4.1 Mahout Bayesian classification Machine Learning a gran escala es otro importante uso de MapReduce. Este test pone a prueba el entrenador Bayesiano ingenuo (un popular algoritmo de clasificación para el descubrimiento de conocimiento y minería de datos) en Mahout0.7, el cual es una librería de código abierto para Machine Learning.

Esta prueba utiliza documentos generados automáticamente cuyas palabras siguen la distribución Zipf. El diccionario utilizado para generar el texto es también el archivo de diccionario por defecto de Linux /usr/share/dict/linux.words.

7.2.4.2 Mahout K-means clustering Este trabajo prueba el agrupamiento Kmeans, un algoritmo de agrupamiento para el descubrimiento de conocimiento y minería de datos. El conjunto de datos de entrada es generado por GenKMeansDataset basado en las distribuciones Continua y Gaussiana.

7.2.5 Data Analytics Benchmarks:

7.2.5.1 Hive Query Benchmarks Esta prueba es desarrollada en base al articulo ”A Comparison of Approaches to Large-Scale Data Analysis” y HIVE-396. Contiene HIVE queries realizando queries típicas OLAP como se describe en el articulo. Su entrada también son datos web generados automáticamente con hiperlinks siguiendo la distribución Zipf.

7.3 Benchmark para Spark

La prueba escogida para poner a prueba el rendimiento de clústers Spark con diferentes cantidades de nodos fue WordCount, de esta manera se puede hacer una comparativa de los rendimientos de ambos frameworks, nuevamente con clústers de distinto tamaño, pasando desde un único nodo hasta contar con 4 nodos. WordCount es un programa representativo de situaciones que se presentan en la vida real, en donde se cuenta con una gran cantidad de datos de la cual se extrae sólo una pequeña parte interesante.

7.4 Desarrollo de las pruebas:

7.4.1 Entregable 1.6: Reporte desempeño y escalabilidad sobre despliegue en Guane.

Como se dijo anteriormente la métrica, sección 7.2, que se tuvo en cuenta es el tiempo que toma el clúster en realizar determinada tarea (tres tomas en total, luego se realizo un promedio). Esta prueba comprende la utilización de clústers con diferente cantidad de nodos comenzando con la utilización de un único nodo en el que se instala todo el software necesario para la correcta ejecución de los Benchmark, el despliegue de los ambientes para crear los clústers sobre Guane necesarios para realizar las pruebas se describen en detalle en el Anexo 3. El entorno hardware utilizado se describe en el Cuadro 3.

Los Benchmark contienen dos partes, el *prepare* que como su nombre lo sugiere prepara los datos para la ejecución de la segunda parte, el *run*, que procesa los datos generados en

el proceso anterior y brinda un resultado final. Se capturaron tres tomas de tiempo para cada Benchmark.

A continuación se muestran los resultados de las pruebas de las que fue posible la ejecución y finalización exitosa de los *jobs:*

7.4.1.1 Resultados para clústers Hadoop - Fase *Prepare*

	PRUEBAS				
NODOS	Sort	WordCount	TeraSort	DFSIOE	PageRank
1	332	331	300	1172	508
2	274	270	153	445	215
3	274	275	105	399	142
4	272	275	92	427	108

Cuadro 4: Primer Toma de Tiempos (s)

	PRUEBAS				
NODOS	Sort	WordCount	TeraSort	DFSIOE	PageRank
1	334	335	302	1172	508
2	275	270	152	457	218
3	273	273	112	398	141
4	270	274	97	417	102

Cuadro 5: Segunda Toma de Tiempos (s)

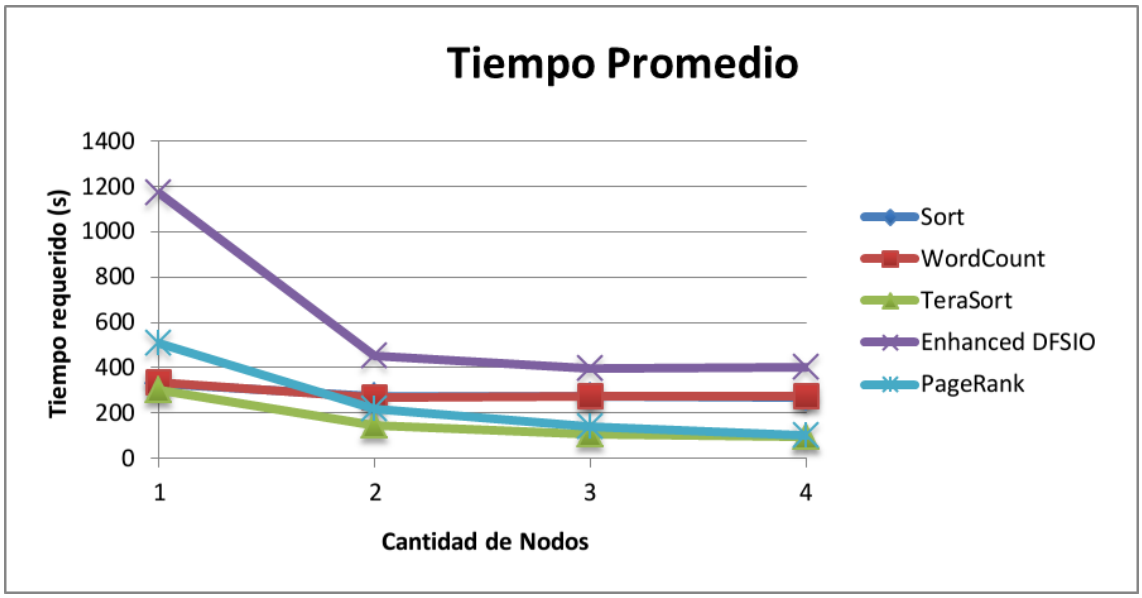
	PRUEBAS				
NODOS	Sort	WordCount	TeraSort	DFSIOE	PageRank
1	333	336	300	1172	508
2	285	276	132	464	225
3	274	273	110	384	138
4	270	274	96	408	99

Cuadro 6: Tercer Toma de Tiempos (s)

	PRUEBAS				
NODOS	Sort	WordCount	TeraSort	DFSIOE	PageRank
1	332.75	333.25	300.5	1172	508
2	277	271.5	147.5	452.75	218.25
3	273.5	274.5	108.5	398.5	141.5
4	271	274.5	94.25	402.25	104.25

Cuadro 7: Tiempo Promedio Fase *Prepare* (s)

Figura 34: Tiempo Promedio Fase Prepare (s)



Fuente: Autores.

El tiempo total empleado para esta fase fue de 5.28 horas.

7.4.1.2 Resultados para clusters Hadoop - Fase *Run*

	PRUEBAS					
NODOS	Sort	WordCount	TeraSort	DFSIOE- Write	DFSIOE- Read	PageRank
1	598	1442	834	1077	1098	996
2	469	1114	345	521	447	403
3	x	1175	232	423	417	272
4	538	1064	186	312	385	200

Cuadro 8: Primer Toma de Tiempos (s)

	PRUEBAS					
NODOS	Sort	WordCount	TeraSort	DFSIOE- Write	DFSIOE- Read	PageRank
1	596	1425	837	1092	1103	994
2	463	1108	342	507	522	402
3	x	1168	233	428	415	270
4	534	1059	185	310	386	203

Cuadro 9: Segunda Toma de Tiempos (s)

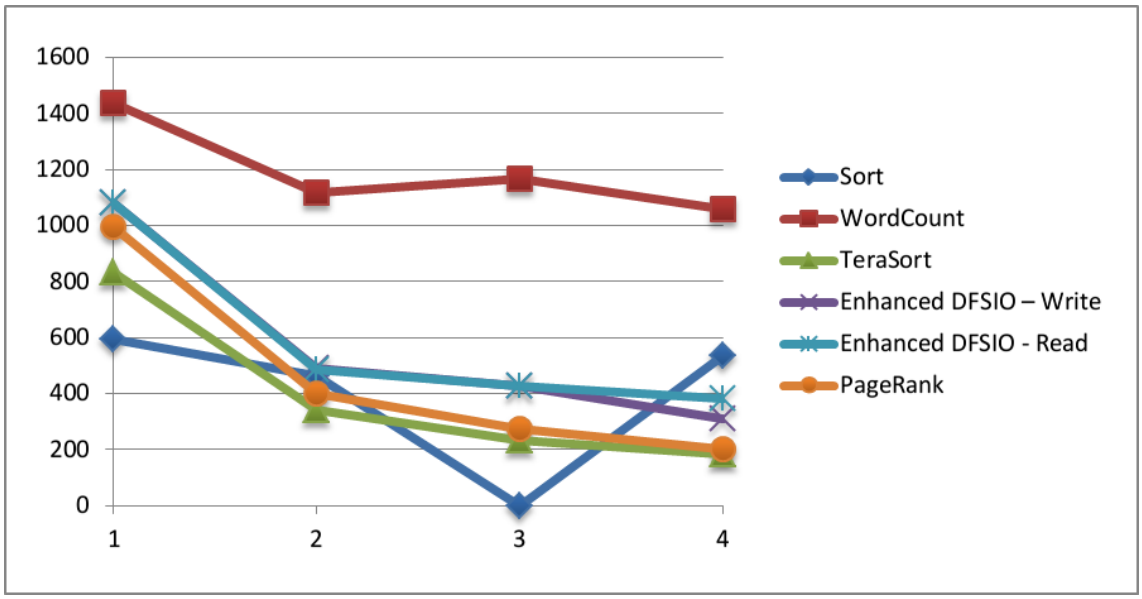
	PRUEBAS					
NODOS	Sort	WordCount	TeraSort	DFSIOE- Write	DFSIOE- Read	PageRank
1	593	1446	834	1075	1041	992
2	461	1129	338	449	486	394
3	x	1156	230	428	452	284
4	533	1059	183	311	375	198

Cuadro 10: Tercer Toma de Tiempos (s)

	PRUEBAS					
NODOS	Sort	WordCount	TeraSort	DFSIOE- Write	DFSIOE- Run	PageRank
1	595.66	1437.66	835	1081.33	1080.66	994
2	464.33	1117	341.66	492.33	485	399.66
3	x	1166.33	231.66	426.33	428	275.33
4	535	1060.66	184.66	311	382	200.33

Cuadro 11: Tiempo Promedio Fase *Run* (s)

Figura 35: Tiempo Promedio Fase Run (s)



Fuente: Autores.

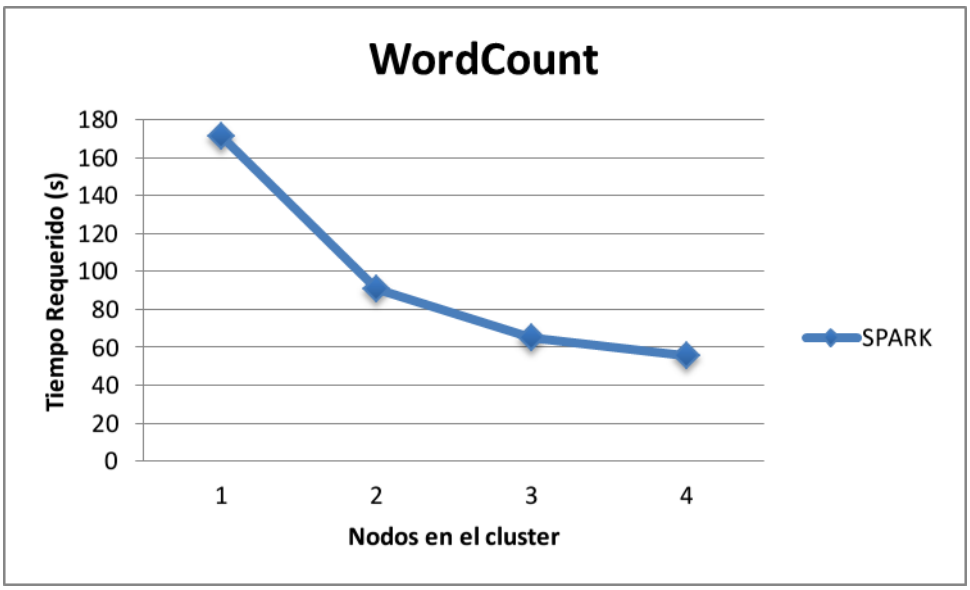
El tiempo total empleado para esta fase fue de 12.1 horas.

7.4.1.3 Resultados para clúster Spark

WordCount - Input 4Gb de texto - Spark					
Nodos	**Toma 1**	**Toma 2**	**Toma 3**	**Toma 4**	**Promedio**
1	172.1007	172.7761	169.8250	170.9221	171.4060
2	93.0177	91.2676	89.6098	89.7382	90.9083
3	68.5719	67.2238	62.8199	60.4996	64.7788
4	57.9533	56.0945	54.4277	53.6268	55.5256

Cuadro 12: Resultados WordCount (s)

Figura 36: Tiempo Promedio Spark



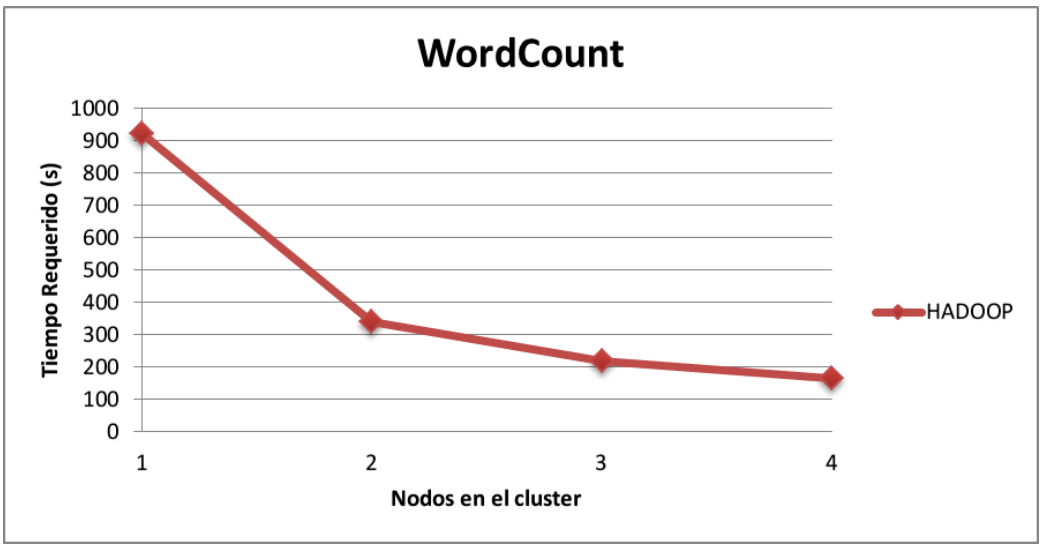
Fuente: Autores.

7.4.1.4 Spark vs Ambari

WordCount - Input 4Gb de texto - Hadoop					
Nodos	**Toma 1**	**Toma 2**	**Toma 3**	**Toma 4**	**Promedio**
1	932	934	916	910	923
2	341	345	339	340	341.25
3	217	217	218	217	217.25
4	162	164	162	165	163.25

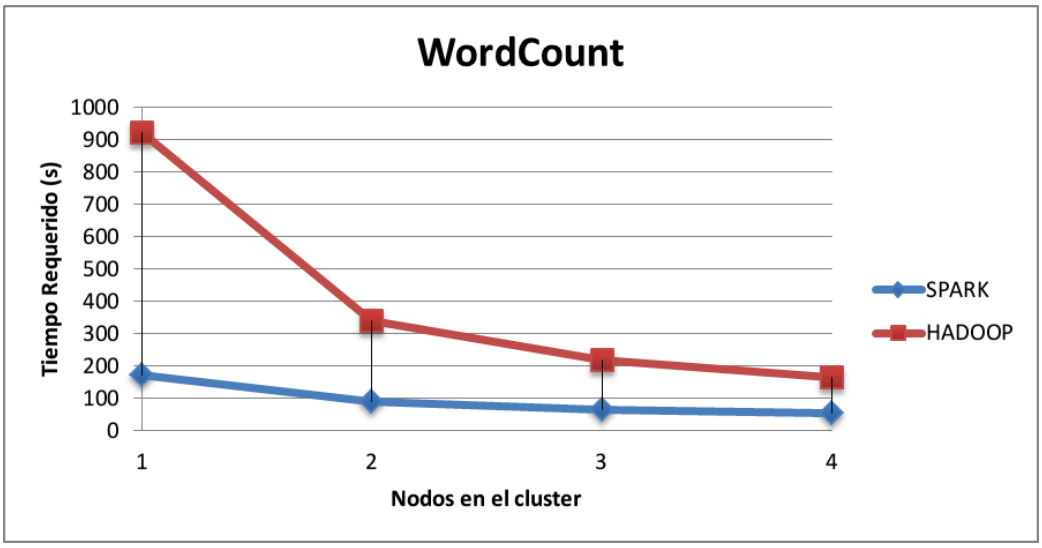
Cuadro 13: Resultados WordCount (s)

Figura 37: Tiempo Promedio Hadoop



Fuente: Autores.

Figura 38: Spark vs Hadoop



Fuente: Autores.

7.5 Análisis de Resultados

7.5.1 Hadoop

Sort es una tarea que transforma los datos de una forma a otra, los datos en el shuffle y datos de salida son del mismo tamaño. Por lo tanto las tareas del tipo sort son tareas orientadas al uso de operaciones de E/S, moderado uso de la CPU. Un caso particular que se puede presentar con estas tareas el "embotellamiento" en la red debido al gran tamaño de los datos en el shufle cuando se corren sobre clústers de gran tamaño. Durante la ejecución de esta prueba hubo un caso muy espacial como se muestra en la figura 35, GUANE03 es agregado como tercer nodo en el clúster, el tamaño de partición asignado para el despliegue con Kadeploy es de 28gb, GUANE08 y GUANE07 por su parte tienen un tamaño de partición asignada de 128gb. Como consecuencia, al generar los datos y posteriormente analizarlos, la capacidad del disco es desbordada llevando a la finalización no exitosa de la prueba. Para resolver este problema se balanceó el clúster pero este proceso simplemente retrasa el resultado fallido.

En el programa Wordcount, cada tarea map da como resultado una dupla *("palabra"-1)* para cada palabra en los datos de entrada, el combinador realiza una suma parcial de cada palabra y finalmente el reduce computa la suma final. Este tipo de tareas extraen una pequeña cantidad de información importante de un gran conjunto de datos, la información del shuffle y los datos finales son mucho mas pequeños que los datos iniciales. Como resultado, este tipo de tareas esta orientado al uso de CPU sobre todo en la etapa map, donde el programa identifica cada palabra en los datos de entrada, obteniendo como resultado un alto consumo de CPU y una utilización ligera del disco, red y operaciones de E/S. Es de esperar que el comportamiento de este tipo de jobs permanezcan mas o menos igual incluso en grandes clústers. En el caso particular de esta prueba se puede observar en el figura 35 que no hay mayor escalabilidad cuando se presentan mas de dos nodos en el clúster. Una posible explicación para este comportamiento es que la generación de datos en esta prueba, al igual que en Sort, es proporcional a la cantidad de nodos presentes en el clúster como se muestra en la figura 34, luego cada host en el clúster tiene mas o menos la misma cantidad de datos para procesar.

Terasort comparte la misma naturaleza de Sort y por lo tanto son orientados al uso de operaciones de E/S y almacenamiento. Sin embargo, este Benchmark comprime los datos salientes de la etapa map para ahorra trafico de red y almacenamiento. Como consecuencia se tiene un uso moderado de la CPU, discos y E/S en la etapa shuffle, mientras que en la etapa reduce tiene un uso moderado de la CPU y alta utilización del disco y E/S.

DFSIO solamente prueba el rendimiento de HDFS, es orientado completamente al uso de disco y operaciones de E/S. Este tipo de tareas usualmente son consumidoras de disco y operaciones de E/S, aunque como en el caso de Terasort se puede comprimir los datos en la etapa shuffle. Este Benchmark trata de mantener los datos en memoria tanto como sea posible.

Finalmente PageRank pasa la mayoría del tiempo realizando iteraciones de diferentes jobs, generalmente pagerank se orienta a la utilización de CPU, con baja o media utilización de memoria y E/S.

7.5.2 Spark

Spark es un poderoso motor de procesamiento de código abierto, construido para ser rápido, fácil de usar y capaz de realizar analística sofisticada. Las aplicaciones Spark corren hasta 100 veces mas rápido en memoria en comparación con Hadoop, y hasta 10 veces mas rápido cuando corre en disco.

Dado que Spark se ejecuta sobre la memoria RAM los tiempos de ejecución son mucho menores, al igual que su interacción con el disco duro que solo se presenta cuando es necesario. En la figura 38, se observa que para jobs del tipo wordcount, Spark es mas eficiente, en términos de desempeño, que Hadoop. Esto también se presenta para otro tipo de jobs gracias a la habilidad de Spark de mantener mas datos residentes en memoria lo que acelera las cargas iterativas.

Para evitar el desbordamiento en la capacidad de almacenamiento se fijó el input en 4gb de datos de tipo texto, dado que las pruebas se realizaron varias veces y en cada iteración tanto Spark como Hadoop realizan la escritura de esta misma cantidad de datos como output, sobre el disco (8 iteraciones x 4 Gb).

8 CONCLUSIONES Y RECOMENDACIONES

- Se genero una curva de aprendizaje en relación al aprendizaje de la configuración de Hadoop y Spark sobre maquinas físicas con repositorios ubicados en la web y locamente. Como resultado de esta curva de aprendizaje se anexan “How To” en los cuales se muestra la configuración de clústers Hadoop y Spark con repositorios en la web y localmente sobre maquinas físicas.

- Se reprodujo la configuración realizada en las maquinas físicas sobre SC3, como resultado, la obtención de las imágenes de despliegue master y worker. Se anexó despliegue sobre Guane de las imágenes master y worker.

- Se definieron casos de uso sobre Hadoop y Spark de analítica de datos que permitieron realizar las pruebas de desempeño y escalabilidad del prototipo de servicio PaaS sobre SC3.

- Se evaluaron los casos definidos para realizar las pruebas del prototipo de servicio PaaS sobre SC3, los cuales mostraron un desempeño adecuado para demostrar la escalabilidad del prototipo de servicio PaaS, como se evidencia en los resultados obtenidos.

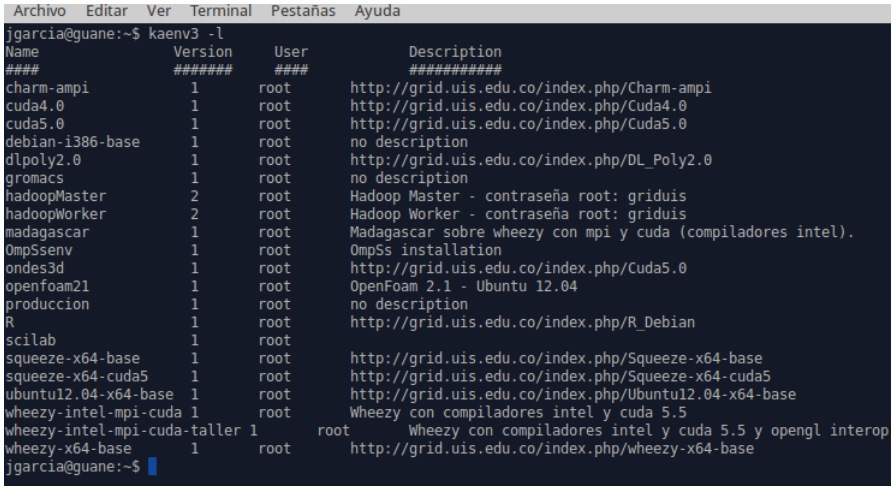
- La capacidad de almacenamiento en disco fue la única falla en términos de desempeño que presentaron los distintos clústers montados sobre SC3. Esta falencia puede ser superada permitiendo el uso de particiones de disco de mayor capacidad asignadas al despliegue con Kadeploy.

- El prototipo de PaaS presenta una arquitectura la cual, por medio de los resultados obtenidos, podemos decir que es escalable.

- Como resultado general del proyecto, se obtuvo el diseño de un prototipo de servicio PaaS sobre OAR / Kadeploy el cual es escalable y además se puede desplegar sobre SC3. El prototipo de servicio PaaS esta conformado por herramientas computacionales para big data tales como Hadoop y Spark.

- Las imágenes para despliegue con kadeploy se encuentran a disposición de investigadores y usuarios en GUANE. Gracias a esta disponibilidad el centro de Supercomputación y Calculo Cientifico UIS (SC3) puede ahora brindar servicios Big Data basados en Hadoop y Spark. Las imágenes para despliegue se encuentran disponibles en GUANE (hadoopMaster, hadoopWorker), como se muestra a continuación:

Figura 39: Disponibilidad Imágenes de Despliegue en SC3-GUANE

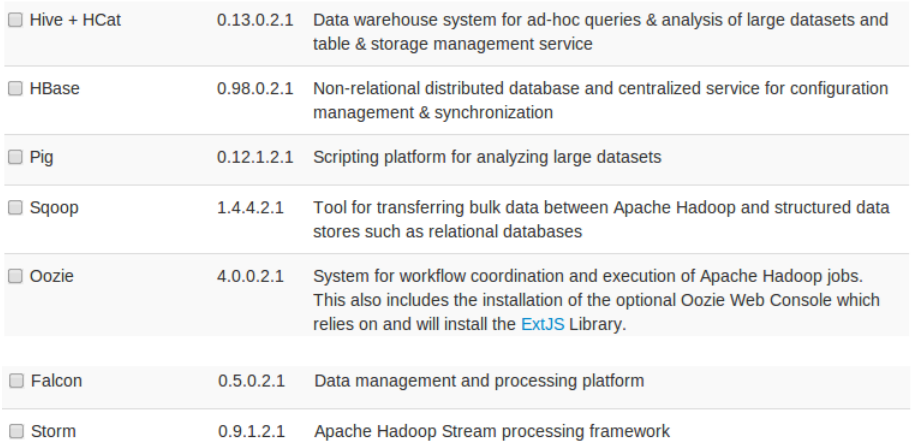


Fuente: Autores.

- Para la realización de las pruebas de desempeño y escalabilidad se utilizaron cuatro nodos de Guane, ya que eran los recursos que se encontraban a disposición para el momento. En un futuro estas pruebas de pueden realizar con mas nodos, inclusive llegar a considerar todo el clúster Guane. Gracias a este proyecto, extender estas pruebas y llevarlas a cabo debería ser relativamente sencillo.

- Para extender el dominio del análisis sobre Big Data hay herramientas que no se tuvieron en cuenta durante el desarrollo del proyecto y que pueden llegar a ser bastante útiles. Herramientas que pueden ser instaladas con Apache Ambari en los clústers que lleguen a ser desplegados sobre Guane:

Figura 40: Herramientas Disponibles



Fuente: Autores.

BIBLIOGRAFÍA

Apache Ambari. (2014, Oct) Apache Ambari-Introduction. [Online]. HYPERLINK. [20]

Apache Hadoop. HDFS Architecture Guide. [Online]. HYPERLINK. [12]

Apache Hadoop. MapReduce Tutorial. [online]. HYPERLINK. [13]

Apache Hadoop. Welcome to Apache™ Hadoop®. [Online]. HYPERLINK. [11]

Ben Lorica. (2012, Nov.) Shark: Real-time queries and analytics for big data. [online]. HYPERLINK. [18]

B. Gentile. (2012, Jun.) Top 5 Myths About Big Data. [Online]. HYPERLINK. [9]

CentOS. CentOS Product Specifications. [Online]. HYPERLINK. [21]

Curso: Sistemas Operativos II Plataforma: Linux-OpenSuse. ”CLUSTER”. [Online]. HYPERLINK. [2]

hortonworks. Apache Ambari. [Online]. HYPERLINK. [19]

“HPC: It’s Not Just for Rocket Scientists Any More”. En: PhD Eadline, Douglas. High Performance Computing For Dummies®, Sun and AMD Special Edition. Indianapolis, Indiana: Wiley Publishing, Inc., Copyright 2009. pp. 3 [1]

IBM.(2012, Jun.) Analytics: el uso de big data en el mundo real. [Online]. HYPERLINK. [10]

“If You Need Speed”. En: PhD Eadline, Douglas. High Performance Computing For Dummies®, Sun and AMD Special Edition. Indianapolis, Indiana: Wiley Publishing, Inc., Copyright 2009. pp. 16 [3]

Kadeploy. Kadeploy - Overview. [Online]. HYPERLINK. [7]

M. Rouse. (2012,) Cloud computing. [Online]. HYPERLINK. [4]

M. Rouse. (2012,) grid computing. [Online]. HYPERLINK. [5]

OAR. (2013, Sep.) What is OAR?. [Online]. HYPERLINK. [6]

Reynold Xin,Joseph Gonzalez, Michael Franklin, Ion Stoica. GraphX: A Resilient Distributed Graph System on Spark [online]. HYPERLINK. [16]

R. Emmett O’Ryan. (2014, Apr.) Apache Spark, la nueva estrella de Big Data. [Online]. HYPERLINK. [14]

R. SchapireLecture. (2008, Feb.) Theoretical Machine Learning. [Online]. HYPERLINK. [8]

spark.apache.org. Machine Learning Library (MLlib). [online]. HYPERLINK. [15]

spark.apache.org. Spark Streaming Programming Guide [online]. HYPERLINK. [17]

Supercomputacion y Calculo Cientifico UIS. Inicio Usuarios-Cómo acceder a la plataforma. [Online]. HYPERLINK. [22]

Wikipedia. (2014, Feb) Universally unique identifier. [Online]. HYPERLINK. [23]

REFERENCIAS BIBLIOGRÁFICAS

Apache Ambari. (2014, Oct) Apache Ambari-Introduction. [Online]. HYPERLINK. [20]

Apache Hadoop. HDFS Architecture Guide. [Online]. HYPERLINK. [12]

Apache Hadoop. MapReduce Tutorial. [online]. HYPERLINK. [13]

Apache Hadoop. Welcome to Apache™ Hadoop®. [Online]. HYPERLINK. [11]

Ben Lorica. (2012, Nov.) Shark: Real-time queries and analytics for big data. [online]. HYPERLINK. [18]

B. Gentile. (2012, Jun.) Top 5 Myths About Big Data. [Online]. HYPERLINK. [9]

CentOS. CentOS Product Specifications. [Online]. HYPERLINK. [21]

Curso: Sistemas Operativos II Plataforma: Linux-OpenSuse. ”CLUSTER”. [Online]. HYPERLINK. [2]

hortonworks. Apache Ambari. [Online]. HYPERLINK. [19]

“HPC: It’s Not Just for Rocket Scientists Any More”. En: PhD Eadline, Douglas. High Performance Computing For Dummies®, Sun and AMD Special Edition. Indianapolis, Indiana: Wiley Publishing, Inc., Copyright 2009. pp. 3 [1]

IBM.(2012, Jun.) Analytics: el uso de big data en el mundo real. [Online]. HYPERLINK. [10]

“If You Need Speed”. En: PhD Eadline, Douglas. High Performance Computing For Dummies®, Sun and AMD Special Edition. Indianapolis, Indiana: Wiley Publishing, Inc., Copyright 2009. pp. 16 [3]

Kadeploy. Kadeploy - Overview. [Online]. HYPERLINK. [7]

M. Rouse. (2012,) Cloud computing. [Online]. HYPERLINK. [4]

M. Rouse. (2012,) grid computing. [Online]. HYPERLINK. [5]

OAR. (2013, Sep.) What is OAR?. [Online]. HYPERLINK. [6]

Reynold Xin,Joseph Gonzalez, Michael Franklin, Ion Stoica. GraphX: A Resilient Distributed Graph System on Spark [online]. HYPERLINK. [16]

R. Emmett O'Ryan. (2014, Apr.) Apache Spark, la nueva estrella de Big Data. [Online]. HYPERLINK. [14]

R. SchapireLecture. (2008, Feb.) Theoretical Machine Learning. [Online]. HYPERLINK. [8]

spark.apache.org. Machine Learning Library (MLlib). [online]. HYPERLINK. [15]

spark.apache.org. Spark Streaming Programming Guide [online]. HYPERLINK. [17]

Supercomputacion y Calculo Cientifico UIS. Inicio Usuarios-Cómo acceder a la plataforma. [Online]. HYPERLINK. [22]

Wikipedia. (2014, Feb) Universally unique identifier. [Online]. HYPERLINK. [23]

ANEXOS

Anexo A: Instalación de cluster Hadoop-Spark utilizando Ambari

INSTALACION CLUSTER HADOOP-SPARK UTILIZANDO AMBARI

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

1. La disposición del cluster es la siguiente:

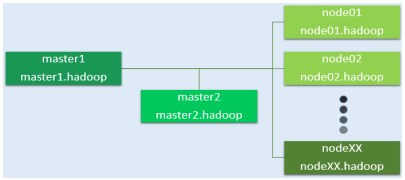


Figura 1– diseño clúster

En la figura 1 se muestra un clúster con un master principal que contiene nuestro servidor ambari además de algunos servicios principales (en este caso master1), un master secundario como soporte para el master principal el cual también contiene algunos servicios principales además de algunos secundarios (en este caso master2) (hay que tener en cuenta que el master2 también puede cumplir funciones de nodo esclavo y tener servicios de cliente, esto en el caso de tener pocos nodos esclavos) y dos nodos esclavos en los cuales se instalan los servicios de cliente (pueden ser tantos nodos como equipos se tengan a disposición).

2. Como prerrequisito se sugiere la instalación de CentOS minimal en ***todos los hosts*** que utilizaran en el clúster

Nota 1: se sugiere que los hostnames de las maquinas sean los siguientes:

Hosts master: master1 y master2

Hosts esclavos: node01, node02 … nodeXX

Y además que la contraseña de usuario root sea la misma para todos los hosts (por comodidad)

Procedemos a ***configurar cada uno de los host*** del cluster:

* Activar puerto de red:

```
#ifup eth0
```

Asumiendo que eth0 es el puerto predeterminado

* Instalar paquetes necesarios:

```
#yum install nano wget openssl ntp
```

Se utilizo nano para la edición de archivos (puede utilizar su editor de preferencia)

* Editar el archivo ifcfg-eth0:

```
#nano /etc/sysconfig/network-scripts/ifcfg-eth0
```

Y cambiar la línea `ONBOOT=no` por `ONBOOT=yes`

* Activar servicio SSH y NTP:

```
#chkconfig sshd on
#chkconfig ntpd on
#ntpdate 0.rhel.pool.ntp.org
```

* Desactivar Iptables:

```
#chkconfig iptables off
#/etc/init.d/iptables stop
```

3. Desde una maquina remota que se encuentre en la misma red del cluster, conectarse atreves de SSH al host el cual se asignara como master1 e ingresar la clave de root asignada

```
$ssh root@<IP.HOST>
```

* Una vez conectado editar el archivo hosts para agregar las IPs, FQDNs y hostnames de cada uno de los hosts que pertenecen al cluster

```
#nano /etc/hosts
```

Por ejemplo (un cluster con dos master y dos nodos esclavos):

```
192.168.66.187 master1.hadoop master1
192.168.66.194 master2.hadoop master2
192.168.66.180 node01.hadoop node01
192.168.66.176 node02.hadoop node02
… … … … … … … … … … … … … … … … … … … …
XXX.XXX.XX.XXX nodeXX.hadoop nodeXX
```

Nota 2: es necesario utilizar un FQDN (nombre de dominio completo) apropiado, el instalador de ambari no reconoce IPs sino nombres de dominio completos, también se recomienda que la maquina de la cual se haga la conexión por SSH contenga además las IPs, FQNDs y hostnames de cada uno de los hosts del cluster (esto para facilitar el monitoreo y conexión)

Sugerencia 1: para evitar la edición de cada fichero `/etc/hosts` en cada host se sugiere enviar el fichero editado en el master1 por SCP a todos los hosts del cluster

```
#scp /etc/hosts root@<hostname>:/etc/hosts
```

* Desactivar SELINUX:

```
#nano /etc/selinux/config
```

Y cambiar la línea `SELINUX=enforcing` por `SELINUX=disabled`

Sugerencia: se debe editar el fichero `/etc/selinux/config` en cada host, para evitar la edición de cada fichero se sugiere enviar el archivo editado en el master1 por SCP a todos los hosts del cluster

```
#scp /etc/selinux/config root@<hostname>:/:/etc/selinux/config
```

4. Reiniciar los servicios de red:

```
#service network restart
```

5. Conectarse de nuevo por SSH al host que se configuro como master1 para generar las llaves SSH e ingresar la clave de root asignada:

```
$ssh root@<IP.HOST>
```

* Generar clave SSH:

```
#ssh-keygen
```

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

* Enviar la clave SSH a ***todos los host*** del cluster:

```
#ssh-copy-id -i ~/.ssh/id_rsa.pub
root@<hostname>

password:XXXXXX
```

6. Instalar servidor ambari (en el host que se asigno como master1):

* Descargar el repositorio:

```
#cd /etc/yum.repos.d/

#wget http://public-repo-
1.hortonworks.com/ambari/centos6/1.x/
updates/1.5.1/ambari.repo
```

* Instalar el servidor ambari:

```
#yum install ambari-server
```

* Configurar el servidor ambari en modo silencioso o predeterminado:

```
#ambari-server setup –s
```

* Iniciar ambari server:

```
#ambari-server start
```

* Reiniciar master1:

```
#reboot
```

7. Instalación de servicios ambari por medio de aplicación web ambari, en un navegador introducir:

```
http://master1.hadoop:8080
```

Esto si cumplió correctamente la ***nota 2,*** si no cumplió con la ***nota 2*** realice la siguiente conexión por medio de la IP del master1:

```
http://192.168.66.187:8080
```

* debe cargar la siguiente pagina de login, Figura 2:

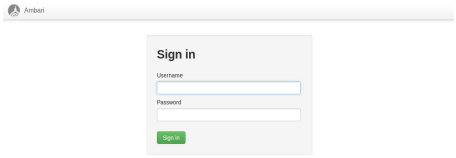


Figura 2

Ambari trae un usuario predeterminado al igual que una contraseña predeterminada:

```
Username: admin

Password: admin
```

* A continuación se muestra el asistente de instalación de cluster, Figura 3:

Como primer paso nombramos nuestro cluster, en este caso lo nombramos `hadoop_cluster`

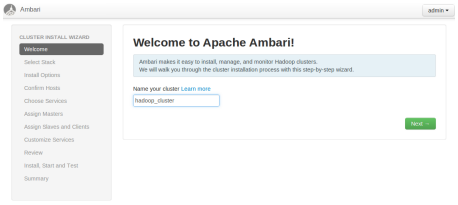


Figura 3

Luego seleccionamos la plataforma de datos mas actualizada de Apache Hadoop, Figura 4:

Seleccionar la ultima versión de la plataforma de datos `HDP 2.1`

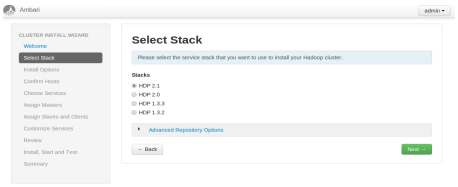


Figura 4

Incluimos la lista de hosts de nuestro cluster usando los FQDNs (nombres de dominio completo), Figura 5, además de la llave privada generada en el **punto 5,** Figura 6:

Para nuestro ejemplo tenemos

```
master1.hadoop

master2.hadoop

node01.hadoop

node02.hadoop
```

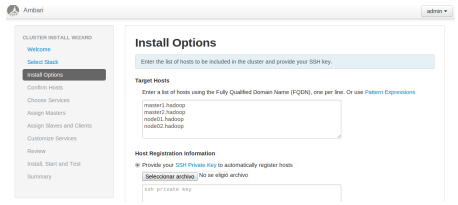


Figura 5

La llave privada se encuentra en el home del master, carpeta .ssh, fichero id_rsa, copiamos la clave privada y la pegamos en el segundo recuadro directamente, Figura 7:

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

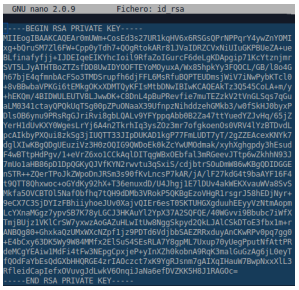


Figura 6

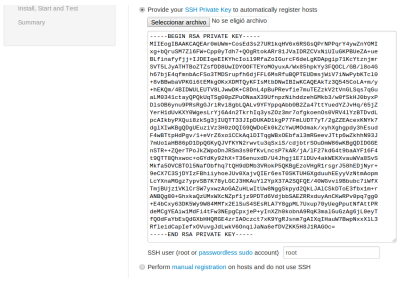


Figura 7

A continuación se realiza el registro de los hosts, Figura 8:

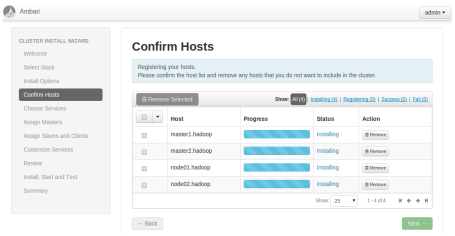


Figura 8

Una ves realizado el registro debe mostrar la siguiente confirmación, Figura 9:

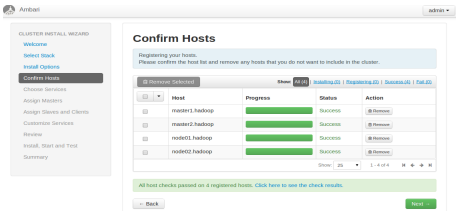


Figura 9

Nota 3: si en este paso falla el registro de hosts se recomienda hacer clic en el botón Retry para reiniciar el proceso; si el registro fallido persiste se recomienda revisar paso a paso los puntos planteados en esta guía para verificar si se realizaron correctamente; llegado el caso de no encontrar solución , es necesario iniciar de nuevo el proceso desde la instalación del sistema operativo CentOS, punto 2.

Luego de realizar el registro de nuestros hosts, se procede a realizar la selección de servicios que se van a instalar en nuestro cluster, Figura 10:

Se recomienda los siguientes servicios como una instalación mínima.

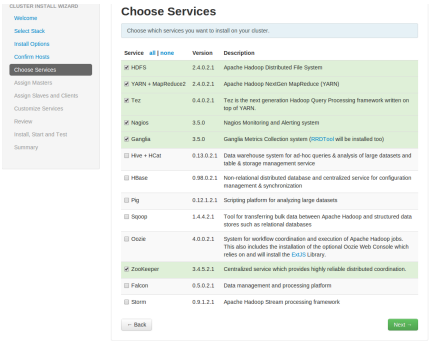


Figura 10

Los servicios adicionales pueden ser instalados luego desde la interfaz web de ambari según sean sus necesidades

A continuación se distribuyen los servicios seleccionados, se recomienda la siguiente distribución, Figura 11:

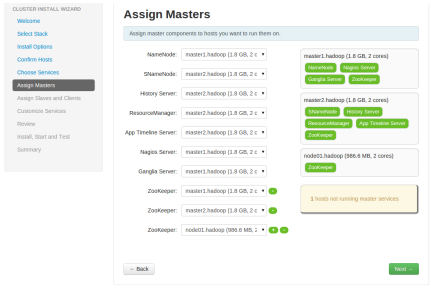


Figura 11

Nota 4: los servicios de HDFS como los son NameNode y SNameNode SIEMPRE DEBEN IR EN EL MASTER PRINCIPAL Y SECUNDARIO RESPECTIVAMENTE.

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

A continuación se realiza la selección de nodos esclavos, Figura 12:

Se seleccionan los hosts que no contienen servicios de master y que contendrán los servicios de cliente y ejecutaran las tareas o jobs

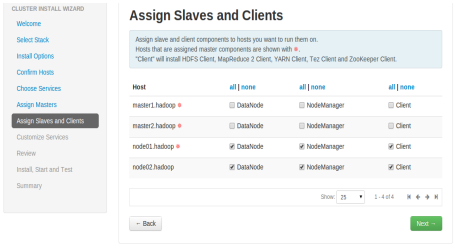


Figura 12

Nota 5: como respaldo para el master principal, Master1, tenemos el master secundario, Master2, el cual además de cumplir esta función es recomendado utilizarlo para realizar tareas de esclavo, depende de su elección.

Como siguiente paso tenemos la personalización de cada uno de los servicios a instalar en el clúster, Figura 13:

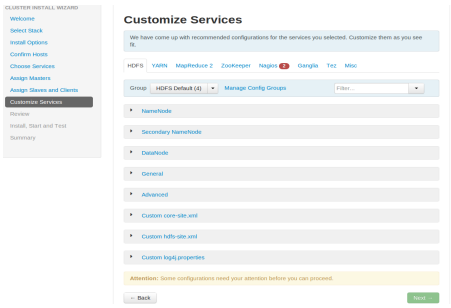


Figura 13

Aunque se sugiere las configuraciones recomendadas por el instalador de ambari; lo único que se debe agregar es el nombre de usuario, contraseña y correo para la administración del servicio Nagios, Figura 14, Figura 15:

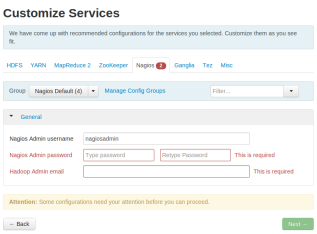


Figura 14

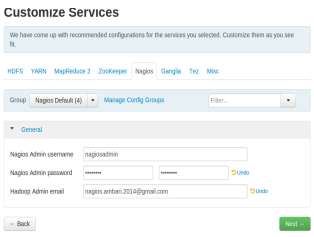


Figura 15

En la pestaña Misc se muestran los usuarios y grupos de usuarios que se crean para cada servicio a instalar en el clúster, Figura 16:

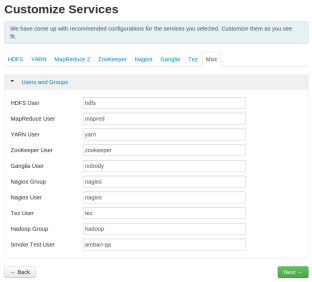


Figura 16

A continuación se genera un archivo de revisión en el cual se describe de una manera general la instalación a ejecutar, como también la ubicación de cada uno de los servicios en el clúster, Figura 17:

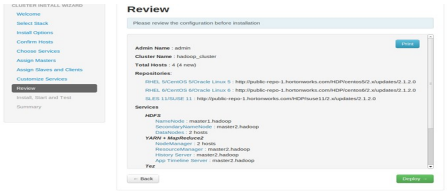


Figura 17

Al dar clic en el botón siguiente se da comienzo a la instalación de los servicios configurados, Figura 18:

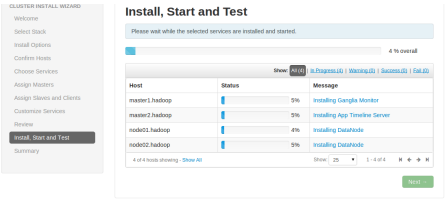


Figura 18

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

Una ves terminada la instalación de todos los servicios en el clúster tenemos una confirmación de instalación, Figura 19:

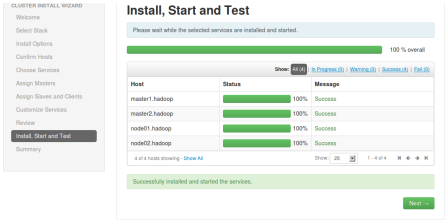


Figura 19

En este punto los servicios ya se encuentran instalados de acuerdo a la distribución configurada anteriormente e iniciados en el clúster

Nota 6: si en este paso falla la instalación de los servicios se recomienda hacer clic en el botón Retry para reiniciar el proceso; llegado el caso de no encontrar solución , es necesario iniciar de nuevo el proceso desde la instalación del sistema operativo CentOS, punto 2.

A continuación se muestra un resumen de instalación, Figura 20:

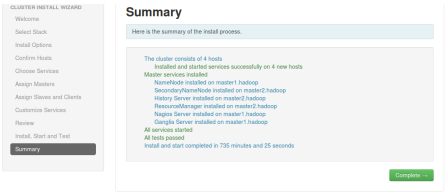


Figura 20

luego se redirige a la interfaz web para la administración y monitoreo del clúster Hadoop, Figura 21:

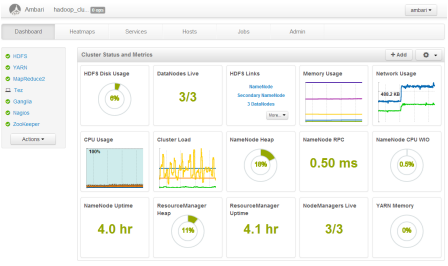


Figura 21

Recomendaciones:

—La contraseña de usuario root debe ser la misma para todos los hosts (por comodidad a la hora de trabajar en la instalación del cluster Hadoop)

—En caso de que en el punto 2 al activar el puerto de red eth0 falle , verificar que el nombre de la interfaz coincida con el de esa tarjeta de red

—Si se desea agregar otro nodo al cluster desde la interfaz web de ambari en la pestaña hosts, se puede agregar un nuevo host esclavo, Figura 22:

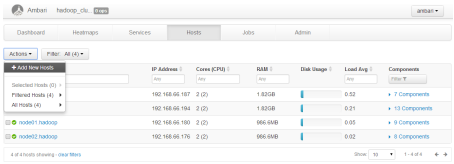


Figura 22

O de igual manera borrar alguno de los hosts, Figura 23:

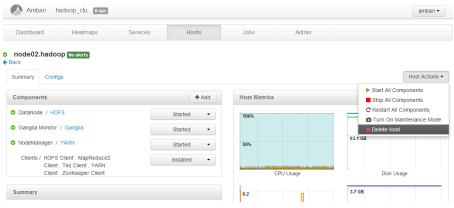


Figura 23

—Al reiniciar el servidor ambari o pararlo, al ingresar de nuevo a la interfaz de administración y monitoreo es necesario iniciar algunos de los servicios manualmente, Figura 24:

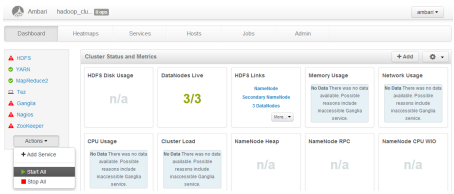


Figura 24

En caso de que algún servicio continúe sin iniciar ir directamente sobre el servicio e iniciarlo independientemente

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

Para la configuración de Spark:

Descargar Spark sobre master1:

```
#wget http://d3kbcqa49mib13.cloudfront.net/spark-1.1.0-bin-hadoop2.4.tgz
```

Descomprimir archivo y mover a la raíz del sistema:

```
#unzip spark-1.1.0-bin-hadoop2.4.tgz -d ~/
#mv spark-1.1.0-bin-hadoop2.4.tgz spark
#mv spark /spark
```

Copiar variables de entorno al usuario HDFS:

```
# cp ~/.bashrc /home/hdfs
```

Agregar Esclavos con hostname:

```
# nano /spark/conf/slaves
```

Enviar archivo slave a los Esclavos:

```
# scp /spark/conf/slaves <hostname>:/spark/conf/slaves
```

Para nuestro cluster ejemplo tenemos:

```
# scp /spark/conf/slaves master1:/spark/conf/slaves

# scp /spark/conf/slaves master2:/spark/conf/slaves

# scp /spark/conf/slaves node01:/spark/conf/slaves

# scp /spark/conf/slaves node02:/spark/conf/slaves
```

Si se tienen mas esclavos se realiza el mismo procedimiento.

Modificar la plantilla spark-env.sh.template:

```
# cp /spark/conf/spark-env.sh.template /spark/conf/spark-env.sh

# nano /spark/conf/spark-env.sh
```

Y agregar al final del archivo la siguiente linea:

```
export SPARK_EXECUTOR_INSTANCES=<num-esclavos>
```

Y se cambia `<num-esclavos>` por el numero de esclavos que se tengan en el clúster, para nuestro clúster ejemplo tenemos:

```
export SPARK_EXECUTOR_INSTANCES=4
```

Los master también puede funcionar como esclavos por eso ponemos 4.

Y por ultimo enviar el archivo recién modificado `spark-env.sh` a los esclavos:

```
# scp /spark/conf/spark-env.sh <hostname>:/spark/conf/spark-env.sh
```

Para nuestro clúster ejemplo tenemos:

```
# scp /spark/conf/spark-env.sh master1:/spark/conf/spark-env.sh

# scp /spark/conf/spark-env.sh master2:/spark/conf/spark-env.sh

# scp /spark/conf/spark-env.sh node01:/spark/conf/spark-env.sh

# scp /spark/conf/spark-env.sh node02:/spark/conf/spark-env.sh
```

Si se tienen mas esclavos se realiza el mismo procedimiento.

Para iniciar la consola Spark en el clúster:

```
# cd /spark
# ./bin/spark-shell --master yarn-client
```

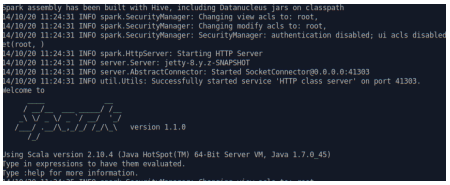


Figura 19– Spark

En este punto tenemos configurado e instalado nuestro cluster Hadoop-Spark.

Anexo B: Instalación de cluster Hadoop-Spark utilizando Ambari con repositorios locales

INSTALACION CLUSTER HADOOP-SPARK UTILIZANDO AMBARI CON REPOSITORIOS LOCALES

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

Universidad Industrial de Santander

CONSTRUIMOS FUTURO

1. La disposición del cluster es la siguiente:

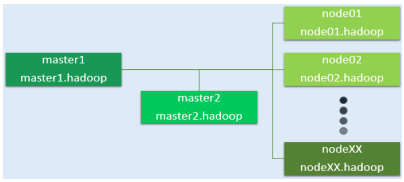


Figura 1– diseño clúster

En la figura 1 se muestra un clúster con un master principal que contiene nuestro servidor ambari además de algunos servicios principales (en este caso master1), un master secundario como soporte para el master principal el cual también contiene algunos servicios principales además de algunos secundarios (en este caso master2)(hay que tener en cuenta que el master2 también puede cumplir funciones de nodo esclavo y tener servicios de cliente, esto en el caso de tener pocos nodos esclavos) y dos nodos esclavos en los cuales se instalan los servicios de cliente (pueden ser tantos nodos como equipos se tengan a disposición).

2. Como prerrequisito se sugiere la instalación de CentOS minimal en ***todos los hosts*** que utilizaran en el clúster

Nota 1: se sugiere que los hostnames de las maquinas sean los siguientes:

Hosts master: master1 y master2

Hosts esclavos: node01, node02 … nodeXX

Y además que la contraseña de usuario root sea la misma para todos los hosts (por comodidad)

Procedemos a ***configurar cada uno de los host*** del cluster:

* Activar puerto de red:

```
#ifup eth0
```

Asumiendo que eth0 es el puerto predeterminado

* Instalar paquetes necesarios:

```
#yum install nano wget openssl ntp
```

Se utilizo nano para la edición de archivos (puede utilizar su editor de preferencia)

* Editar el archivo ifcfg-eth0:

```
#nano /etc/sysconfig/network-scripts/ifcfg-eth0
```

Y cambiar la línea `ONBOOT=no` por `ONBOOT=yes`

* Activar servicio SSH y NTP:

```
#chkconfig sshd on
#chkconfig ntpd on
#ntpdate 0.rhel.pool.ntp.org
```

* Desactivar Iptables:

```
#chkconfig iptables off
#/etc/init.d/iptables stop
```

3. Desde una maquina remota que se encuentre en la misma red del cluster, conectarse atreves de SSH al host el cual se asignara como master1 e ingresar la clave de root asignada

```
$ssh root@<IP.HOST>
```

* Una vez conectado editar el archivo hosts para agregar las IPs, FQDNs y hostnames de cada uno de los hosts que pertenecen al cluster

```
#nano /etc/hosts
```

Por ejemplo (un cluster con dos master y dos nodos esclavos):

```
192.168.66.187 master1.hadoop master1
192.168.66.194 master2.hadoop master2
192.168.66.180 node01.hadoop node01
192.168.66.176 node02.hadoop node02
… … … … … … … … … … … … … … … … … … … …
XXX.XXX.XX.XXX nodeXX.hadoop nodeXX
```

Nota 2: es necesario utilizar un FQDN (nombre de dominio completo) apropiado, el instalador de ambari no reconoce IPs sino nombres de dominio completos, también se recomienda que la maquina de la cual se haga la conexión por SSH contenga además las IPs, FQNDs y hostnames de cada uno de los hosts del cluster (esto para facilitar el monitoreo y conexión)

Sugerencia 1: para evitar la edición de cada fichero `/etc/hosts` en cada host se sugiere enviar el fichero editado en el master1 por SCP a todos los hosts del cluster

```
#scp /etc/hosts root@<hostname>:/etc/hosts
```

* Desactivar SELINUX:

```
#nano /etc/selinux/config
```

Y cambiar la línea `SELINUX=enforcing` por `SELINUX=disabled`

Sugerencia: se debe editar el fichero `/etc/selinux/config` en cada host, para evitar la edición de cada fichero se sugiere enviar el archivo editado en el master1 por SCP a todos los hosts del cluster

```
#scp /etc/selinux/config root@<hostname>:/:/etc/selinux/config
```

4. Reiniciar los servicios de red:

```
#service network restart
#reboot
```

5. Conectarse de nuevo por SSH al host que se configuro como master1 para generar las llaves SSH e ingresar la clave de root asignada:

```
$ssh root@<IP.HOST>
```

* Generar clave SSH:

```
#ssh-keygen
```

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

Universidad Industrial de Santander
CONSTRUIMOS FUTURO

* Enviar la clave SSH a ***todos los host*** del cluster:

```
#ssh-copy-id -i ~/.ssh/id_rsa.pub
root@<hostname>

password:XXXXXX
```

6. Instalar repo de ambari y HDP:

Instalar los .repo de ambari y HDP en:

```
#cd /etc/yum.repos.d/

#wget http://public-repo-1.hortonworks.com/
ambari/centos6/1.x/updates/1.6.1/
ambari.repo

#wget http://s3.amazonaws.com/public-
repo-1.hortonworks.com/HDP/
centos6/2.x/2.1-latest/hdp.repo
```

Y actualizar:

```
repo id              repo name                                        status
HDP-2.1              HDP                                                  80
HDP-2.1.4.0          Hortonworks Data Platform Version - HDP-2.1.4.0      80
HDP-UTILS-1.1.0.17   HDP-UTILS                                            48
Updates-ambari-1.6.1 ambari-1.6.1 - Updates                                5
ambari-1.x           Ambari 1.x                                            5
base                 CentOS-6 - Base                                   6.367
extras               CentOS-6 - Extras                                    14
updates              CentOS-6 - Updates                                1.362
repolist: 7.961
```

Figura 2– lista de repositorios

```
#yum update
```

Para listar los repositorios disponibles en el sistema:

```
#yum repolist
```

Figura 2– repositorios disponibles

7. Crear repositorios locales de Ambari:

De los repositorios que se muestran en la Figura 2, nos interesan, Updates-ambari-1.6.1 y ambari-1.x

Crear un directorio en el cual se almacenaran los paquetes de los repos:

```
#mkdir /repos
```

* Utilizamos reposync para traer los paquetes de los repos:

```
#reposync –r <repo-id> -p <dir-to-save>
```

Como ejemplo:

```
#reposync –r Updates-ambari-1.6.1 –p /repos

#reposync –r ambari-1.x –p /repos
```

* Una vez traídos los paquetes es necesario recrear el metadata del repo para ser reconocido por yum:

```
#createrepo /repos/Updates-ambari-1.6.1/

#createrepo /repos/ambari-1.z/
```

8. Modificar los .repo en /etc/yum.repos.d/

```
#nano /etc/yum.repos.d/ambari.repo
```

* Y cambiar las urls remotas en _baseurl por las adecuadas URLs del servidor local, por ejemplo:

```
baseurl=http://public-repo-
1.hortonworks.com/ambari/centos6/1.x/GA
```

Lo cambiamos por:

```
baseurl=http://node01/shared/ambari-1.x/
```

Actualizar

```
#yum update
```

En este punto los repos ya están disponibles para la red local

9. Crear el servidor local:

```
#cd /var/www/html

#ln –s /var/www/html/ /repos
```

Con esto se crea un enlace simbolico entre la carpeta donde se encuentran los repos y la carpeta que alberga los archivos del servidor local.

10. Instalar ambari-server:

Esta instalación ya funciona con los repos locales.

```
#yum install ambari-server
```

Descargar jdk-7u45-linux-x64.tar.gz y guardar el paquete en `/var/lib/ambari-server/resources`

```
#wget http://public-repo-1.hortonworks.com/
ARTIFACTS/jdk-7u45-linux-x64.tar.gz
```

Correr `#ambari-server setup –s` y automáticamente instala jdk en `/usr/jdk64/`

11. instalacion de JAVA usando Alternatives:

```
#cd /usr/jdk64/jdk1.7.0_45

#alternatives –install /usr/bin/java java /
usr/jdk64/jdk1.7.0_45/bin/java 2

#alternatives –config java
```

Seleccionar la opción adecuada

```
#alternatives –install /usr/bin/jar jar /
usr/jdk64/jdk1.7.0_45/bin/jar 2

#alternatives –install /usr/bin/javac ja-
vac /usr/jdk64/jdk1.7.0_45/bin/javac 2

#alternatives –set jar /usr/jdk64/
jdk1.7.0_45/bin/jar

#alternatives –set javac /usr/jdk64/
jdk1.7.0_45/bin/javac
```

Compruebe versión de JAVA:

```
#java –versión
```

Variables de entorno de instalacion

```
#export JAVA_HOME=/usr/jdk64/jdk1.7.0_45

#export JRE_HOME=/usr/jdk64/jdk1.7.0_45/jre

#export PATH=$PATH:/usr/jdk65/jdk1.7.0_45/
bin:/usr/jdk64/jdk1.7.0_45/jre/bin
```

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

* Instalar el servidor ambari:

```
#yum install ambari-server
```

* Configurar el servidor ambari en modo silencioso o predeterminado:

```
#ambari-server setup -s
```

* Iniciar ambari server:

```
#ambari-server start
```

* Reiniciar master1:

```
#reboot
```

7. Instalación de servicios ambari por medio de aplicación web ambari, en un navegador introducir:

```
http://master1.hadoop:8080
```

Esto si cumplió correctamente la ***nota 2,*** si no cumplió con la ***nota 2*** realice la siguiente conexión por medio de la IP del master1:

```
http://192.168.66.187:8080
```

* debe cargar la siguiente pagina de login, Figura 3:

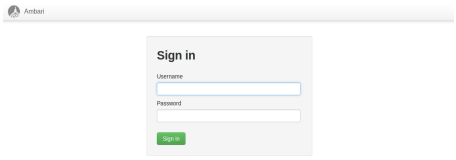


Figura 3

Ambari trae un usuario predeterminado al igual que una contraseña predeterminada:

```
Username: admin
```

```
Password: admin
```

* A continuación se muestra el asistente de instalación de cluster, Figura 4:

Como primer paso nombramos nuestro cluster, en este caso lo nombramos `hadoop_cluster`

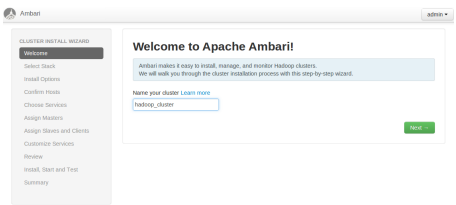


Figura 4

Luego seleccionamos la plataforma de datos mas actualizada de Apache Hadoop, Figura 5:

Seleccionar la ultima versión de la plataforma de datos `HDP 2.1`

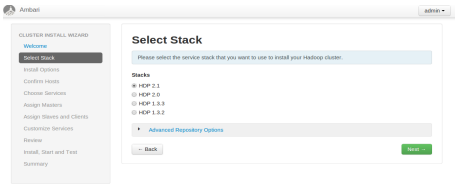


Figura 5

Y luego en opciones avanzadas cambiamos las direcciones de los repositorios remotos por las direcciones almacenadas localmente, puntos 6, 7, 8 y 9, Figura 6:

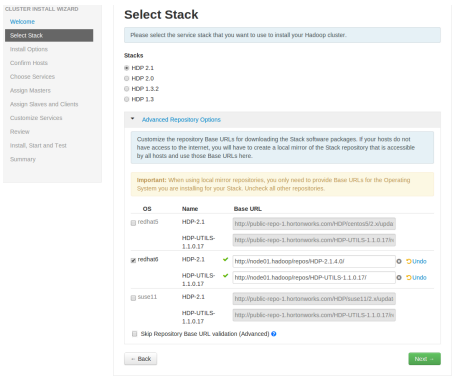


Figura 6

Luego incluimos la lista de hosts de nuestro cluster usando los FQDNs (nombres de dominio completo), Figura 7:

Por ejemplo tenemos:

```
master1.hadoop
```

```
master2.hadoop
```

```
node01.hadoop
```

```
node02.hadoop
```

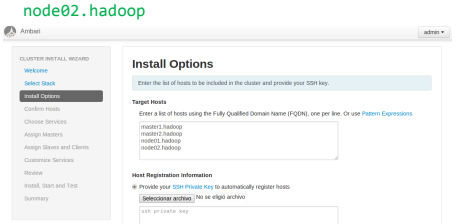


Figura 7

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

además de la llave privada generada en el **punto 5,** Figura 8:

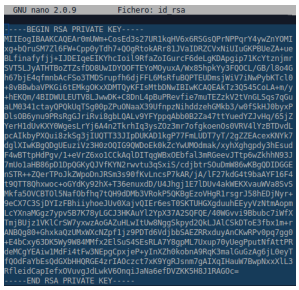


Figura 8

La llave privada se encuentra en el home del master, carpeta .ssh, fichero id_rsa, copiamos la clave privada y la pegamos en el segundo recuadro directamente, Figura 9:

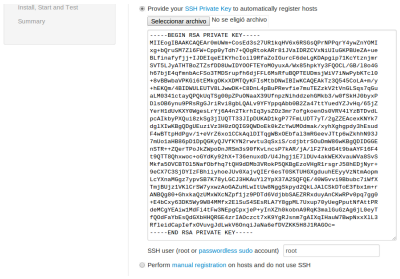


Figura 9

A continuación se realiza el registro de los hosts, Figura 10:

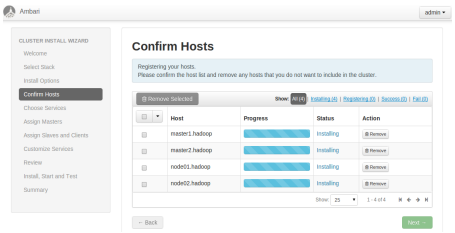


Figura 10

Una ves realizado el registro debe mostrar la siguiente confirmación, Figura 11:

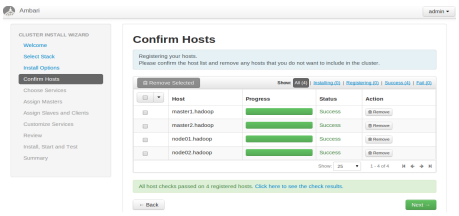


Figura 11

Nota 3: si en este paso falla el registro de hosts se recomienda hacer clic en el botón Retry para reiniciar el proceso; si el registro fallido persiste se recomienda revisar paso a paso los puntos planteados en esta guía para verificar si se realizaron correctamente; llegado el caso de no encontrar solución , es necesario iniciar de nuevo el proceso desde la instalación del sistema operativo CentOS, punto 2.

Luego de realizar el registro de nuestros hosts, se procede a realizar la selección de servicios que se van a instalar en nuestro cluster, Figura 12:

Se recomienda los siguientes servicios como una instalación mínima.

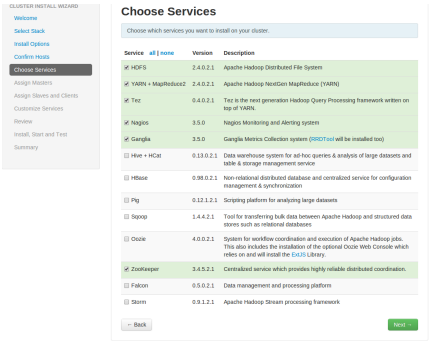


Figura 12

Los servicios adicionales pueden ser instalados luego desde la interfaz web de ambari según sean sus necesidades

A continuación se distribuyen los servicios seleccionados, se recomienda la siguiente distribución, Figura 13:

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

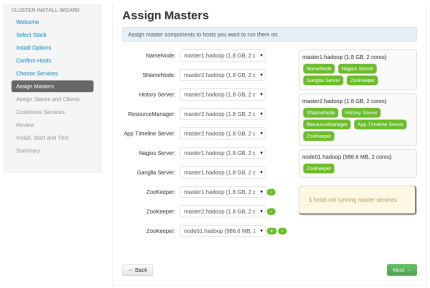


Figura 13

Nota 4: los servicios de HDFS como los son NameNode y SNameNode SIEMPRE DEBEN IR EN EL MASTER PRINCIPAL Y SECUNDARIO RESPECTIVAMENTE.

A continuación se realiza la selección de nodos esclavos, Figura 14:

Se seleccionan los hosts que no contienen servicios de master y que contendrán los servicios de cliente y ejecutaran las tareas o Jobs

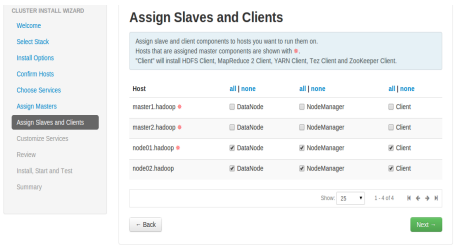


Figura 14

Nota 5: como respaldo para el master principal, Master1, tenemos el master secundario, Master2, el cual además de cumplir esta función es recomendado utilizarlo para realizar tareas de esclavo, depende de su elección.

Como siguiente paso tenemos la personalización de cada uno de los servicios a instalar en el clúster, Figura 15:

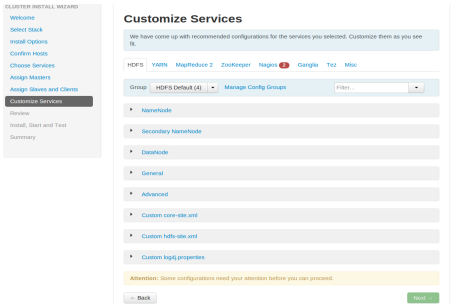


Figura 15

Aunque se sugiere las configuraciones recomendadas por el instalador de ambari; lo único que se debe agregar es el nombre de usuario, contraseña y correo para la administración del servicio Nagios, Figura 16, Figura 17:

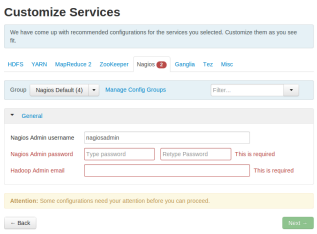


Figura 16

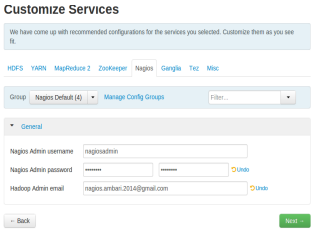


Figura 17

En la pestaña Misc se muestran los usuarios y grupos de usuarios que se crean para cada servicio a instalar en el clúster, Figura 18:

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

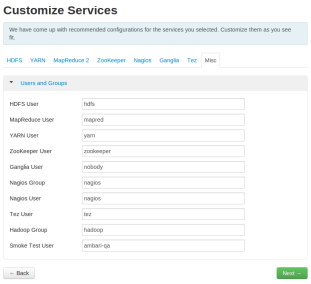


Figura 18

A continuación se genera un archivo de revisión en el cual se describe de una manera general la instalación a ejecutar, como también la ubicación de cada uno de los servicios en el clúster, Figura 19:

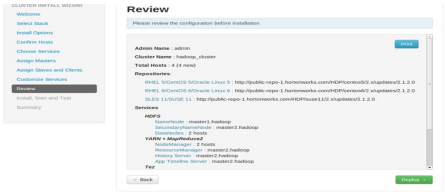


Figura 19

Al dar clic en el botón siguiente se da comienzo a la instalación de los servicios configurados, Figura 20:

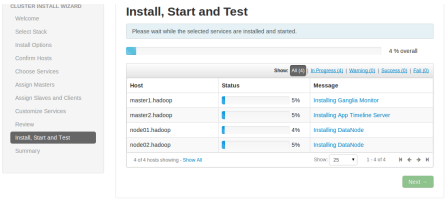


Figura 20

Una ves terminada la instalación de todos los servicios en el clúster tenemos una confirmación de instalación, Figura 21:

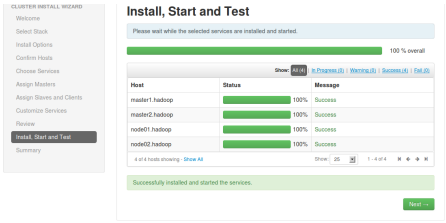


Figura 21

En este punto los servicios ya se encuentran instalados de acuerdo a la distribución configurada anteriormente e iniciados en el clúster

Nota 6: si en este paso falla la instalación de los servicios se recomienda hacer clic en el botón Retry para reiniciar el proceso; llegado el caso de no encontrar solución , es necesario iniciar de nuevo el proceso desde la instalación del sistema operativo CentOS, punto 2.

A continuación se muestra un resumen de instalación, Figura 22:

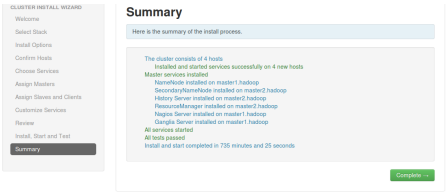


Figura 22

luego se redirige a la interfaz web para la administración y monitoreo del clúster Hadoop, Figura 23:

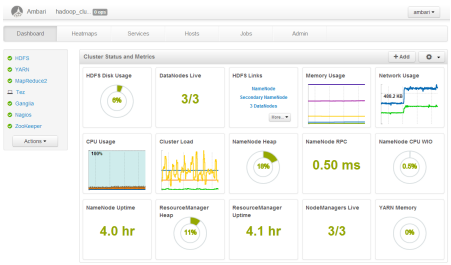


Figura 23

<u>Recomendaciones:</u>

—La contraseña de usuario root debe ser la misma para todos los hosts (por comodidad a la hora de trabajar en la instalación del cluster Hadoop)

—En caso de que en el punto 2 al activar el puerto de red eth0 falle , verificar que el nombre de la interfaz coincida con el de esa tarjeta de red

—Si se desea agregar otro nodo al cluster desde la interfaz web de ambari en la pestaña hosts, se puede agregar un nuevo host esclavo, Figura 24:

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

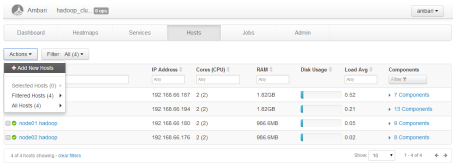


Figura 24

O de igual manera borrar alguno de los hosts, Figura 25:

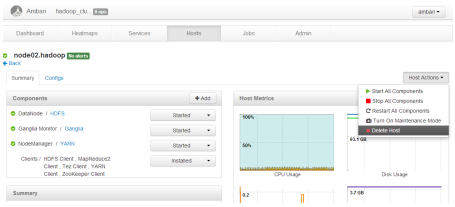


Figura 25

—Al reiniciar el servidor ambari o pararlo, al ingresar de nuevo a la interfaz de administración y monitoreo es necesario iniciar algunos de los servicios manualmente, Figura 26:

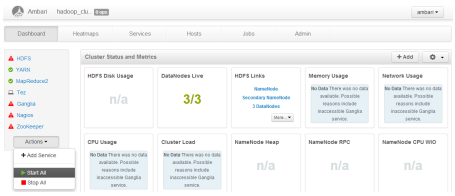


Figura 26

En caso de que algún servicio continúe sin iniciar ir directamente sobre el servicio e iniciarlo independientemente

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

Para la configuración de Spark:

Descargar Spark sobre master1:

```
#wget http://d3kbcqa49mib13.cloudfront.net/
spark-1.1.0-bin-hadoop2.4.tgz
```

Descomprimir archivo y mover a la raíz del sistema:

```
#unzip spark-1.1.0-bin-hadoop2.4.tgz -d ~/
#mv spark-1.1.0-bin-hadoop2.4.tgz spark
#mv spark /spark
```

Copiar variables de entorno al usuario HDFS:

```
# cp ~/.bashrc /home/hdfs
```

Agregar Esclavos con hostname:

```
# nano /spark/conf/slaves
```

Enviar archivo slave a los Esclavos:

```
# scp /spark/conf/slaves <hostname>:/spark/
conf/slaves
```

Para nuestro cluster ejemplo tenemos:

```
# scp /spark/conf/slaves master1:/spark/conf/
slaves
```

```
# scp /spark/conf/slaves master2:/spark/conf/
slaves
```

```
# scp /spark/conf/slaves node01:/spark/conf/
slaves
```

```
# scp /spark/conf/slaves node02:/spark/conf/
slaves
```

Si se tienen mas esclavos se realiza el mismo procedimiento.

Modificar la plantilla spark-env.sh.template:

```
# cp /spark/conf/spark-env.sh.template /spark/
conf/spark-env.sh
```

```
# nano /spark/conf/spark-env.sh
```

Y agregar al final del archivo la siguiente linea:

```
export SPARK_EXECUTOR_INSTANCES=<num-esclavos>
```

Y se cambia `<num-esclavos>` por el numero de esclavos que se tengan en el clúster, para nuestro clúster ejemplo tenemos:

```
export SPARK_EXECUTOR_INSTANCES=4
```

Los master también puede funcionar como esclavos por eso ponemos 4.

Y por ultimo enviar el archivo recién modificado `spark-env.sh` a los esclavos:

```
# scp /spark/conf/spark-env.sh <hostname>:/
spark/conf/spark-env.sh
```

Para nuestro clúster ejemplo tenemos:

```
# scp /spark/conf/spark-env.sh master1:/spark/
conf/spark-env.sh
```

```
# scp /spark/conf/spark-env.sh master2:/spark/
conf/spark-env.sh
```

```
# scp /spark/conf/spark-env.sh node01:/spark/
conf/spark-env.sh
```

```
# scp /spark/conf/spark-env.sh node02:/spark/
conf/spark-env.sh
```

Si se tienen mas esclavos se realiza el mismo procedimiento.

Para iniciar la consola Spark en el clúster:

```
# cd /spark
# ./bin/spark-shell --master yarn-client
```

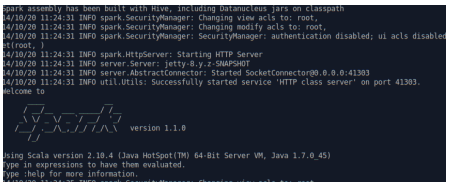


Figura 19– Spark

En este punto tenemos configurado e instalado nuestro cluster Hadoop-Spark.

Anexo C: Instalación de cluster Hadoop-Spark utilizando Ambari sobre SC3-Guane

INSTALACION CLUSTER HADOOP-SPARK UTILIZANDO AMBARI-SOBRE SC3-GUANE

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

1– Como primer paso para la instalación de un Clúster Hadoop-Spark se debe tener acceso a los recursos computacionales del SC3, para esto se debe solicitar una cuenta en:

http://usuarios.grid.uis.edu.co/index.php/Inicio_Usuarios

2- Una vez asignada la cuenta de usuario, por medio de una consola accedemos al servidor `toctoc.grid.uis.edu.co` a través del protocolo de SSH(secure shell):

```
# ssh <usuario>@toctoc.grid.uis.edu.co
```

Ingresamos nuestra contraseña.

3- Una vez dentro de la plataforma, ingresamos al servidor guane, desde donde podrá realizar reservas en los diferentes clústers disponibles.

```
# ssh guane
```

Ingresamos nuestra contraseña.

4– Verificamos sobre la herramienta web Monika los nodos de Guane que estén disponibles para realizar el despliegue:

```
http://guane.uis.edu.co/monika
```

5– Se realiza la reserva de los nodos que estén disponibles sobre Guane:

Para este despliegue utilizamos los nodos guane04 y guane05 para disponer de un clúster de 2 nodos (master-worker).

```
#oarsub -t deploy -l walltime=300 -p "network_address='guane04'" -r "2014-09-04 15:30:00"
```

```
#oarsub -t deploy -l walltime=300 -p "network_address='guane05'" -r "2014-09-04 15:30:00"
```

6– Listamos la base de datos de ambientes disponibles donde se encuentran las imágenes de despliegue master y worker.

```
#kaenv3 -l
```

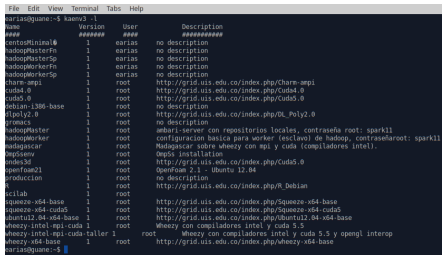


Figura 1- Ambientes Disponibles Sobre Guane

7– Realizamos el despliegue sobre las maquinas reservadas en Guane:

Sobre la maquina guane04 desplegamos la imagen de despliegue Master:

```
#kadeploy3 –e hadoopMasterSp –m guane04
```

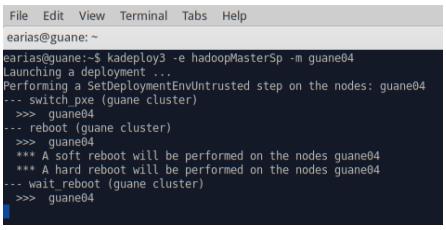


Figura 2– Despliegue Imagen Master

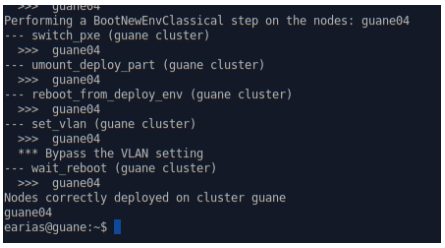


Figura 3– Confirmación Despliegue Imagen Master

Sobre la maquina guane04 desplegamos la imagen de despliegue Worker:

```
#kadeploy3 –e hadoopWorkerSp –m guane05
```

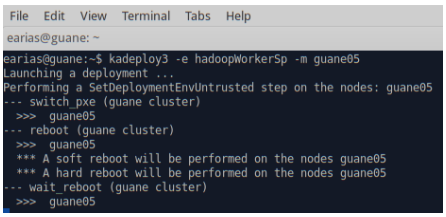


Figura 4- Despliegue Imagen Worker

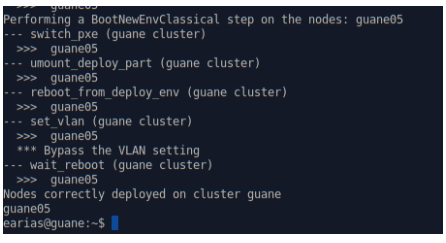


Figura 5- Confirmación Despliegue Imagen Worker

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

7– Ingresar por ssh al master (en este caso guane04):

```
#ssh root@guane04
```

```
Ingresar clave: “griduis”
```

8– Modificar el archivo hosts:

```
#nano /etc/hosts
```

E ingresar las ips correspondientes de los nodos que conforman le clúster:

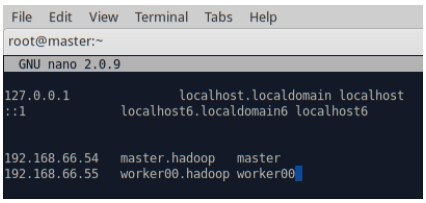


Figura 6– Nombre de Host e Ips

Y enviarlo via ssh a todos los worker:

```
scp /etc/hosts root@<hostname>:/etc/host
```

9– Instalamos Ambari Server:

```
#yum install ambari-server
```

Y se realiza la configuración automática del servidor con el siguiente comando:

```
#ambari-server setup –s
```

E iniciar el servicio:

```
#ambari-server start
```

10– Generar clave SSH:

```
#ssh-keygen
```

Enviar la clave SSH publica a ***todos los host*** del clúster:

En este caso solo tenemos el host master y worker00

```
#ssh-copy-id -i ~/.ssh/id_rsa.pub root@<hostname>
```

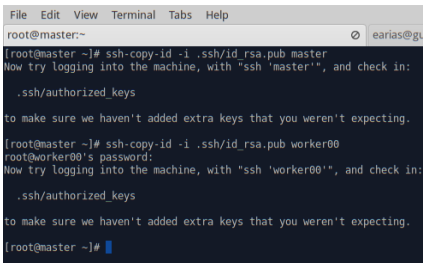


Figura 7– Envió de Llave Publica

11- Instalación de servicios ambari por medio de aplicación web ambari, en un navegador introducir la IP de la maquina en al cual se despliega la imagen Master:

```
http://192.168.66.54:8080
```

Debe cargar la siguiente pagina de login, Figura 8:

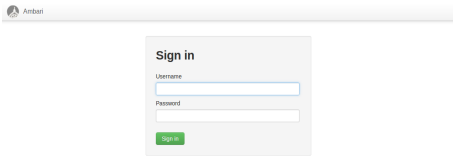


Figura 8– Login

Ambari trae un usuario predeterminado al igual que una contra-seña predeterminada:

```
Username: admin
```

```
Password: admin
```

A continuación se muestra el asistente de instalación de cluster, Figura 9:

Como primer paso nombramos nuestro cluster, en este caso lo nombramos `hadoop_cluster`

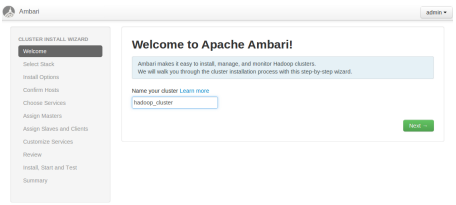


Figura 9– Nombre Clúster

Luego seleccionamos la plataforma de datos mas actualizada de Apache Hadoop, y agregamos las URL`s de los repositorios locales:

Las URL`s de los repositorios locales se encuentran en el Master:

```
http://192.168.66.54/repos/HDP-UTILS-1.1.0.19/
```

```
http://192.168.66.54/repos/HDP-2.1.5.0/
```

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

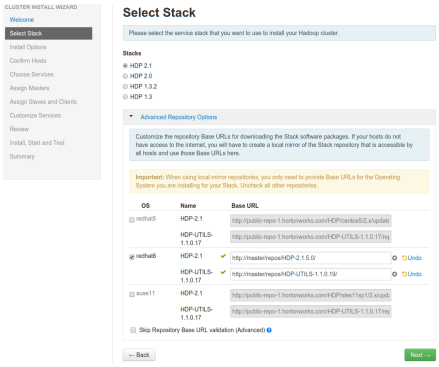


Figura 10– Repos Locales

12- Luego incluimos la lista de hosts de nuestro clúster usando los FQDNs (nombres de dominio completo) y la llave SSh privada (id_rsa):

`master.hadoop`

`worker00.hadoop`

…….

`workerXX.hadoop`

Y asi tantos worker`s como sea necesario.

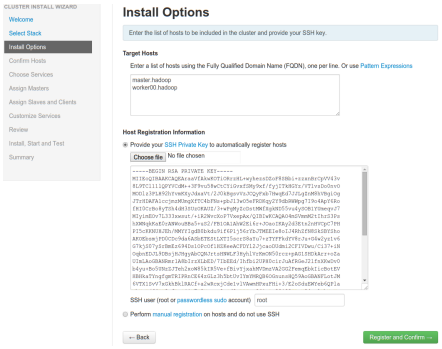


Figura 11– Host y Llave Privada

La llave privada se encuentra en el home del master, carpeta .ssh, fichero id_rsa.

13- A continuación se realiza el registro de los hosts, y una vez realizado el registro debe mostrar la siguiente confirmación:

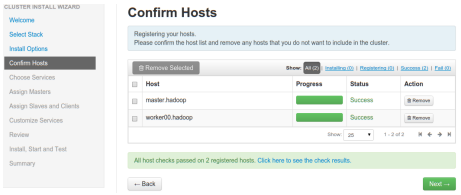


Figura 12– Registro de Hosts

14- Luego de realizar el registro de nuestros hosts, se procede a realizar la selección de servicios que se van a instalar en nuestro clúster:

Se recomienda los siguientes servicios como una instalación para utilizar Hadoop-MapReduce además del monitoreo para el clúster.

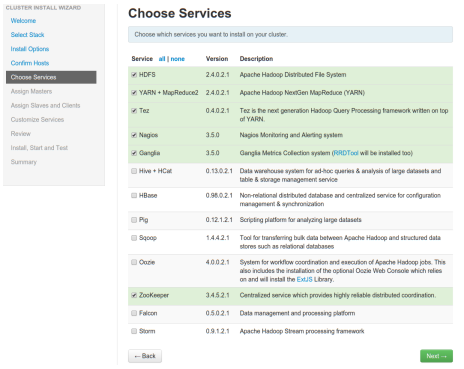


Figura 13– Servicios Seleccionados

15- A continuación se distribuyen los servicios seleccionados, se recomienda la siguiente distribución:

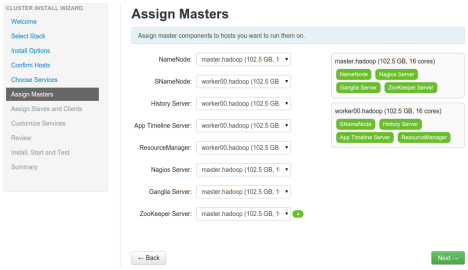


Figura 14– Distribución de Servicios

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

16- A continuación se realiza la selección de DataNodes y servicios de cliente, nodos worker:

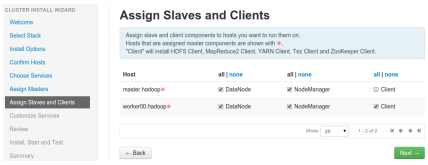


Figura 15– Selección de DataNodes-Servicios Clienteç

17- Como siguiente paso tenemos la personalización de cada uno de los servicios a instalar en el clúster, aunque se recomienda las configuraciones estándar. Ingresar campos requeridos para Nagios:

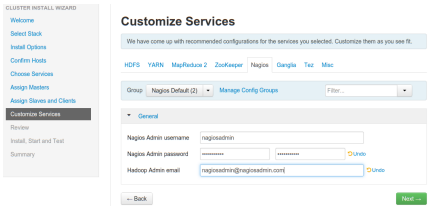


Figura 16– Configuración Nagios

18- En la pestaña Misc se muestran los usuarios y grupos de usuarios que se crean para cada servicio a instalar en el clúster, en la pestaña Review, se genera un archivo de revisión en el cual se describe de una manera general la instalación a ejecutar, como también la ubicación de cada uno de los servicios en el clúster.

19– Luego se instalan todos los servicios configurados en el clúster:

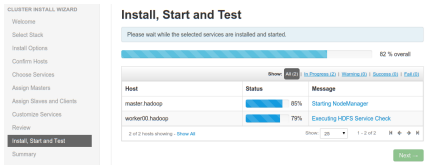


Figura 17– Progreso Instalación

Luego aparece una confirmación de instalación exitosa:

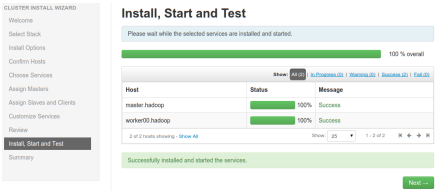


Figura 18– Confirmación Instalación

En este punto se tiene isntalado un cluster Hadoop funcional, como siguiente paso configurar Spark:

20– Para la configuración de Spark:

Copiar variables de entorno al usuario HDFS:

```
# cp ~/.bashrc /home/hdfs
```

Agregar Wokers con hostname:

```
# nano /spark/conf/slaves
```

Enviar archivo slave a los workers:

```
# scp /spark/conf/slaves <workers>:/spark/conf/slaves
```

Para nuestro cluster ejemplo tenemos:

```
# scp /spark/conf/slaves worker00:/spark/conf/slaves
```

Si se tienen mas workers se realiza el mismo procedimiento.

Modificar la plantilla spark-env.sh.template:

```
# cp /spark/conf/spark-env.sh.template /spark/conf/spark-env.sh
```

```
# nano /spark/conf/spark-env.sh
```

Y agregar al final del archivo la siguiente linea:

```
export SPARK_EXECUTOR_INSTANCES=<num-workers>
```

Y se cambia `<num-workers>` por el numero de Workers que se tengan en el clúster, para nuestro clúster ejemplo tenemos:

```
export SPARK_EXECUTOR_INSTANCES=2
```

El master también puede funcionar como worker por eso ponemos 2.

Y por ultimo enviar el archivo recién modificado `spark-env.sh` a los workers:

```
# scp /spark/conf/spark-env.sh <hostaname>:/spark/conf/spark-env.sh
```

Josymar Garcia Acevedo, Juan Eduardo Arias Valero, Escuela de Ingeniería de Sistemas, Universidad Industrial de Santander, Bucaramanga-Colombia

Para nuestro clúster ejemplo tenemos:

```
# scp /spark/conf/spark-env.sh worker00:/spark/
conf/spark-env.sh
```

Si se tienen mas workers se realiza el mismo procedimiento.

Para iniciar la consola Spark en el clúster:

```
# cd /spark
# ./bin/spark-shell --master yarn-client
```

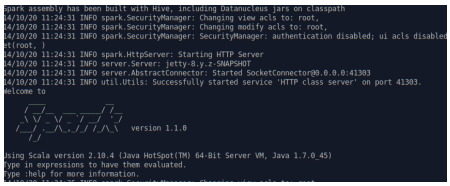


Figura 19– Spark

En este punto tenemos configurado e instalado nuestro cluster Hadoop-Spark.

NOTA 1:

A la hora de agregar mas nodos worker al cluster Hadoop-Spark se debe cambiar el hostname de la maquina con el ambiente desplegado para evitar conflictos de comunicación ya que la imagen de despliegue trae por defecto el hostname worker00:

Editar el archivo:

```
#nano /etc/sysconfig/network
```

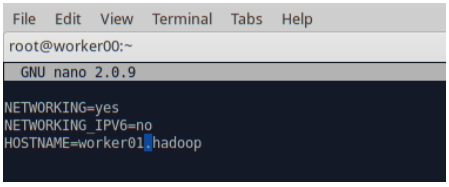


Figura20– Cambio de Hostname