DISEÑO DE UNA ARQUITECTURA BIG DATA PARA LA PREDICCIÓN DE CRISIS EN EL TRASTORNO BIPOLAR

Julio César Anchiraico Trujillo

MÁSTER EN INGENIERÍA INFORMÁTICA, FACULTAD DE INFORMÁTICA,
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin Máster en Ingeniería Informática

Madrid, febrero de 2017

Directora:

Dra. María Victoria López López

Autorización de Difusión

El abajo firmante, matriculado en el Máster en Ingeniería Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "Diseño de una Arquitectura Big Data para la predicción de crisis en el Trastorno Bipolar", realizado durante el curso académico 2016-2017 bajo la dirección de María Victoria López López en el Departamento de Arquitectura de Computadores y Automática, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Autor: Julio César Anchiraico Trujillo
Fecha: Madrid, febrero de 2017

Resumen

El trastorno bipolar conduce, muchas veces, a períodos de baja por enfermedad y la necesidad a una atención muy cercana, generando problemas económicos, laborales, sociales y familiares. La crisis, en la mayoría de los pacientes, se puede evitar mediante la predicción temprana. El proyecto "Bip4Cast" de la Clínica Nuestra Señora de la Paz (Orden Hospitalaria de San Juan de Dios) y del grupo GRASIA-GTeC de la Universidad Complutense de Madrid, propone el diseño de una arquitectura Big Data para la generación, adquisición, almacenamiento y análisis de los datos de diversas fuentes con el objetivo de predecir las crisis del trastorno bipolar. Este Trabajo de Fin de Máster es el análisis de este tipo de arquitecturas desde un punto de vista crítico. En el trabajo se han analizado y comparado distintas opciones de arquitectura del software y se ha implementado la alternativa óptima para la resolución de los problemas derivados de un entorno Big Data en el que se enmarca el proyecto. Al no existir estándares reconocidos se analizaron las propuestas más cercanas a la normalización por lo que se escogió la propuesta del Grupo de trabajo de Big Data del Instituto Nacional de Estándares y Tecnología de los Estados Unidos. Se propone un prototipo de arquitectura con la distribución Hadoop Hortonworks donde fueron implementadas aplicaciones y herramientas según las características del problema. Las pruebas realizadas en la fase de experimentación demuestran que la arquitectura cumple con el tratamiento óptimo de los datos tanto en su diversidad, seguridad, inmutabilidad y flexibilidad, otorgando al proyecto las bases necesarias para alcanzar la fase de análisis.

Palabras clave

Big Data, Hadoop, Análisis Predictivo, Trastorno Bipolar, Actigrafía, HDFS, Arquitectura

Abstract

Bipolar disorder often leads to periods of sick leave and a close attention, generating economic, labor, social and family problems. These crises, in most patients, are avoidable across the early prediction. This work forms part of the project “Bip4cast”. It is a developed between the Clinic of Nuestra Señora de La Paz (Orden Hospitalaria de San Juan de Dios) and the G-Tec group of the Complutense University of Madrid. This work proposes the design of Big Data Architecture whose aim is to implement the generation, acquisition, storage and analysis of data from various sources to predict bipolar disorder crises.

To carry out the implementation, is necessary the use of the ecosystem Hadoop, which integrates platforms that allow the storage and distributed processing of large data volumes, which makes this platform ideal for implementing architectures that can exploit the features of Big Data.

There are distributions of Hadoop that implement most of the platforms that are part of the ecosystem, in addition to their own tools. In the proposed architecture, has been implemented distribution Hadoop Hortonworks that gives us various applications and tools in the different layers of a structure Big Data. This architecture has been implemented and configured according to the characteristics of the problem.

The tests realized in the phase of experimentation demonstrate that the Architecture fulfills with the ideal treatment of the information so much in his diversity, safety, immutability and flexibility. These features grant to the project the bases necessary for the aim in predicting crises in bipolar disorder.

Keywords

Big Data, Hadoop, Bipolar Disorder, Predictive Analysis, Actigraphy, Hadoop, HDFS, Architecture

Índice General

Índice de Figuras

Índice de Tablas

Agradecimientos

A mi directora María Victoria López, por darme la oportunidad de desarrollar el presente trabajo, por su tiempo y recursos de investigación que hicieron posible el logro de los objetivos planteados.

Mi más sincero agradecimiento a mi familia, en especial, a mis padres y hermana, quienes con su apoyo incondicional han permitido culminar mis estudios con una gran satisfacción.

Finalmente, agradezco a la Universidad Complutense de Madrid por haberme brindado la oportunidad de vivir una nueva experiencia profesional y personal.

Capítulo 1. Introducción

El trastorno bipolar en sus distintas formas afecta al 2,4% de la población mundial [1]. Esta enfermedad maníaco-depresiva, es un trastorno cerebral que causa cambios inusuales en el estado de ánimo, niveles de actividad, y la capacidad para llevar a cabo las tareas diarias [2]. Las personas que lo sufren tienen un alto riesgo de suicidio, cerca al 20% [3], incluso con tratamiento más del 30% de los pacientes sufrirá al menos una recaída en el primer año tras su diagnóstico y más del 60% tienen una nueva crisis en los dos primeros años. Se inicia, clásicamente, durante la adolescencia o primeros años de la edad adulta y afecta, a la persona, a lo largo de toda su vida.

El tratamiento farmacológico es fundamental en el abordaje de la enfermedad. Su principal objetivo es acortar las crisis y prevenir su aparición, pero la medicación tiene importantes efectos secundarios, sobre todo, a dosis elevadas. Considerando estos antecedentes en el tratamiento de la enfermedad, es de vital importancia su detección temprana. El tratamiento inmediato ante una nueva crisis puede suponer una gran diferencia en la efectividad del mismo. Sin embargo, esta detección precoz es muy difícil, ya que al principio de una crisis los síntomas pueden ser muy sutiles, casi imposibles de reconocer tanto para el paciente como para la familia.

En las crisis del desorden bipolar el común denominador es el cambio en el patrón de sueño y de la actividad física, que se manifiestan semanas antes de una crisis. De realizar la detección temprana de estos cambios, se permitiría tratamientos más eficaces, se mejoraría la posibilidad de integración socio-laboral de los pacientes y se disminuiría las dosis en el tratamiento farmacológico necesario para su estabilización.

El proyecto “Bip4Cast” de la Clínica Nuestra Señora de La Paz (Orden Hospitalaria de San Juan de Dios) y el grupo GRASIA-GTeC de la Universidad Complutense de Madrid, busca predecir mediante técnicas como la actigrafía [4] (una técnica médica que permite la monitorización continuada del sueño y la actividad física) la aparición de crisis en pacientes con desorden bipolar. Además, se cuenta con otras fuentes de datos como el historial clínico de cada paciente, evaluaciones periódicas del médico y una aplicación móvil que consiste en un cuestionario para el seguimiento diario del paciente. La combinación de toda esta información y su procesamiento implementa un sistema de alarma que notifica al paciente y a su médico de la proximidad de la crisis, permitiendo tomar las medidas adecuadas para evitar su aparición. Por lo

que se ha clasificado al proyecto como Big Data principalmente por el volumen de datos que tiene que procesar, la variedad de fuentes y otras características que son detalladas en la sección 4.2.1.

Definido el problema, se planea implementar una solución Big Data y como primer paso se diseñará una arquitectura que brinde soporte a la generación, adquisición, almacenamiento y análisis de datos. En el presente trabajo se tomó como referencia la propuesta del [5]. Esta implementación estará en un entorno Hadoop [6] que es una plataforma diseñada para el almacenamiento y procesamiento distribuido de datos, cuenta con escalabilidad horizontal, es tolerante a fallos gracias al sistema de archivos distribuido HDFS [7] que replica automáticamente los bloques de datos a nodos distintos, además, de ser de código abierto que funciona bajo la licencia de la Fundación de Software Apache [6] [8].

Existen una variedad de distribuciones Hadoop [9] que integran herramientas que se especializan en cada capa de una arquitectura Big Data, como es el caso de la distribución Hortonworks que es de código abierto [10] a diferencia de otras distribuciones como Cloudera o MapR. Con la utilización de una distribución Hortonworks es posible implementar una arquitectura Big Data de menor coste y tiempo. Considerando dichos beneficios esta ha sido la elegida para la implementación de la arquitectura propuesta.

1.1. Objetivo del proyecto

El objetivo del presente trabajo es proponer una arquitectura Big Data que considere la generación, adquisición, almacenamiento y análisis de datos para dar soporte a la predicción de crisis en pacientes diagnosticados con trastorno bipolar.

La arquitectura propuesta plantea la utilización y elección adecuada de diversas tecnologías en los diferentes niveles de una estructura Big Data acorde con las características del problema. Además, de que pueda procesar diversas fuentes de datos entre las que destacamos: el historial clínico, las evaluaciones periódicas del médico, respuestas a cuestionarios diarios hechos al paciente desde una aplicación móvil acerca de consumo de drogas, hora de acostarse, etc; y principalmente datos relacionados con la actividad física y el sueño capturados en actígrafos [4].

Así mismo, se plantea realizar pruebas del funcionamiento de la arquitectura en el tratamiento de la principal fuente de datos (actígrafos) desde su recolección hasta su procesamiento.

1.2. Trabajos relacionados

El descubrimiento de patrones y comportamientos a partir de volúmenes de datos de diversas fuentes ya sea para obtener resultados en tiempo real o para análisis posteriores, viene aplicándose en diversas áreas de investigación y comercialización, como son: salud, redes sociales, seguridad, mercadotecnia, finanzas, entre otras.

Una arquitectura Big Data debe ser diseñada según las necesidades del problema o investigación a abordar donde se evalúa el tipo de fuentes de datos: estructuradas, semi-estructuradas, no estructuradas, la necesidad del procesamiento de los datos: si este será en tiempo real o no (procesamiento *batch*), la privacidad: si es necesario aplicar técnicas de anonimización o no, etc. En el artículo *"Big Data Architecture for Pervasive Healthcare"* [11] se establecen los componentes principales de una arquitectura Big Data dentro de la asistencia sanitaria generalizada que implica la recopilación de datos a través de dispositivos móviles y redes de sensores, donde los datos son, por lo general, de grandes volúmenes, diversos y de alta frecuencia. La naturaleza de Big Data, volumen, variedad y velocidad [12], junto a su capacidad analítica complementan la prestación de la asistencia sanitaria generalizada. Este estudio ofrece tres contribuciones importantes: 1) identifica los temas de investigación de Big Data y la asistencia sanitaria generalizada, 2) establece la relación entre los temas de investigación, que más tarde forman parte de la arquitectura de Big Data para la asistencia sanitaria generalizada, y 3) propone una arquitectura Big Data general para proyectos de asistencia sanitaria generalizada.

Una solución Big Data puede ser implementada dentro de una organización utilizando su propia infraestructura, pero esto conlleva una inversión importante que, en muchos casos, no es posible asumir. Después de evaluar la cantidad de datos que serán recolectados, almacenados y procesados, y, de estimar la cantidad de procesamiento necesario para obtener los resultados, se podrá establecer si es conveniente externalizar toda o parte de la arquitectura Big Data. Actualmente se cuenta con diversos servicios en la nube, en el artículo *"Big Data Storage Architecture Design in Cloud Computing"* [13] se plantea resolver el problema que representa el almacenamiento tradicional y la gestión de datos de grandes volúmenes, con una plataforma construida sobre Hadoop en una infraestructura local y otra para el almacenamiento de datos en la nube. Con el fin de romper los cuellos de botella del nodo maestro, principalmente relacionados con la actualización de datos, se propone un sistema de almacenamiento de mejor rendimiento alojado en la nube que entre otras ventajas presenta a la recuperación de fallos que ayuda

enormemente en la gestión del sistema de almacenamiento (cuello de botella común en la actualización de datos).

Como se ha descrito hasta el momento, existen una diversidad de propuestas de arquitecturas para la implementación de soluciones Big Data, por lo que surge la necesidad de definir un estándar para los componentes de la arquitectura y los modelos operacionales que en conjunto dan a lugar al denominado ecosistema Big Data. En el artículo *"Defining architecture components of the Big Data Ecosystem"* [14] se analizan los diversos ámbitos de dominio de Big Data, en la ciencia, industria y actividades sociales. El artículo propone una definición de Big Data a partir de sus propiedades o características, que incluyen: volumen, velocidad, variedad, valor y veracidad; además de modelos de datos y estructuras, analítica de datos, infraestructura y seguridad. La arquitectura propuesta (*Big Data Architecture Framework*) incluye los siguientes componentes: infraestructura, analítica, modelos y estructuras de datos, gestión del ciclo de vida y seguridad.

El Instituto Nacional de Estándares y Tecnología (NIST por sus siglas en inglés [15]) es una agencia del Departamento de Comercio de los Estados Unidos que trabaja en establecer un consenso sobre los conceptos fundamentales de Big Data. Esta agencia ha creado el *NIST Big Data Public Working Group (NBD-PWG),* un grupo de trabajo encargado de elaborar una serie de volúmenes denominados *NIST Big Data Interoperability Framework*. En la presente investigación se ha tomado como referencia el volumen 6 [5], que resume el trabajo realizado por el *NBD-PWG* para caracterizar a Big Data desde una perspectiva basada en la arquitectura, presenta el modelo conceptual de una referencia de arquitectura de Big Data *(NIST Big Data Reference Architecture - NBDRA)* y se analizan los componentes de la NBDRA. Todo esto nos brinda un punto de partida para definir nuestra propia arquitectura para el proyecto Bip4Cast.

En cuanto a la utilización de los actígrafos como fuentes de datos se han encontrado diversas investigaciones que han hecho uso de los mismos. En el artículo *"Wrist actigraphy"* [4] se define a la actigrafía como una herramienta útil para el estudio del sueño, que permite su evaluación durante periodos de tiempo prolongados. Se presenta a la actigrafía como una estimación válida del TST *(Total Sleep Time),* el porcentaje de sueño y WASO *(Wake After Sleep Onset).* Aunque esta técnica no puede reemplazar a otras herramientas de evaluación, tales como entrevistas clínicas o diarios de sueño, puede proporcionar información útil en la evaluación del insomnio y los trastornos del sueño de ritmo circadiano. El estudio del sueño es preponderante en

la evaluación del patrón de un paciente diagnosticado con trastorno bipolar, por ello, los datos obtenidos por los actígrafos son la fuente principal dentro de la arquitectura propuesta.

En el artículo *"A Novel, Open Access Method to Assess Sleep Duration Using a Wrist-Worn Accelerometer"* [16] se utilizaron los datos generados por los actígrafos para elaborar un algoritmo de alcance general que evalúe el sueño de 60-83 participantes que llevaban un acelerómetro durante 9 días consecutivos. Este algoritmo establece que una persona se encuentra en un periodo de sueño al identificar cambios en el ángulo del brazo mayores a 5 grados durante 5 minutos o más, lo cual es registrado como periodo de sueño del participante.

La detección de patrones de comportamiento para definir estados de sueño, sedentarismo y estar en actividad fueron evaluados en el artículo *"Behavioral Periodicity Detection from 24 h Wrist Accelerometry and Associations with Cardiometabolic Risk and Health-Related Quality of Life"* [17] mediante la observación de periodicidades (repetición de patrones). Los datos generados por el actígrafo pueden capturar patrones sin explotar, incluyendo el sueño, sedentarismo y comportamientos activos. El actígrafo utilizado para este estudio es el GENEActiv [18], el mismo que es usado por el proyecto Bip4Cast. Los resultados demuestran que mediante la captura de datos con un actígrafo es posible caracterizar los patrones de comportamiento validados por otros métodos formales en el ámbito de la salud.

Los artículos antes mencionados hacen referencia a la utilización de actígrafos para la captura de datos, permitiendo obtener patrones de sueño y actividad física. Este referente ha sido considerado como parte del componente de generación de datos dentro de la arquitectura Big Data propuesta.

1.3. Estructura del trabajo

El presente trabajo se divide en 6 capítulos:

El capítulo 2 describe los planteamientos de las arquitecturas para Big Data y los conceptos de los elementos participantes en cada capa, además de los casos de uso, requisitos de Big Data y modelo conceptual de la arquitectura propuesta por el NIST.

El capítulo 3 explora el ecosistema Hadoop y las herramientas que ofrece para la implementación de un entorno Big Data, desde la ingesta de datos hasta el análisis y visualización de resultados. También se realiza un análisis de las características más importantes en las principales distribuciones Hadoop y una comparativa entre estas.

El capítulo 4 detalla la arquitectura Big Data propuesta por este Trabajo de Fin de Máster para el desarrollo del proyecto Bip4Cast. Se explican los detalles de una arquitectura que aborde la generación, recolección, almacenamiento y análisis de los datos necesarios, definiendo las tecnologías y herramientas adecuadas según las características del proyecto.

El capítulo 5 detalla las implementaciones y configuraciones realizadas de las diversas tecnologías definidas en el modelo de arquitectura propuesto para el proyecto.

El capítulo 6 presenta las pruebas y experimentos realizados desde la captura de datos con el servicio SFTP hasta el procesamiento dentro de la distribución Hadoop Hortonworks para comprobar el funcionamiento de la arquitectura propuesta.

Finalmente, en el capítulo 7 se presentan las conclusiones y las líneas de trabajo futuro para el proyecto.

Chapter 1. Introduction

Bipolar disorder in its different forms affects 2.4% of the world population [1]. It is also known as manic-depressive disease, is a brain disorder that causes unusual changes in mood, activity levels, and the ability to carry out daily tasks [2]. People who suffer it have a high risk of suicide about 20 % [3], even with treatment over 30% of patients suffer at least one relapse in the first year after diagnosis and more than 60% have a new crisis in the first two years. It begins during adolescence or adulthood first years, affecting the person throughout his life. Pharmacological treatment is fundamental in the approach of the disease. The main goal is to shorten the crisis and prevent its occurrence; but medication has significant side effects, especially at high doses. These are necessary for the treatment of crises. Considering the precedents treatment of the disease is of vital importance to detect, as soon as possible, the onset of a crisis. Immediate treatment before a new crisis can make a big difference in the effectiveness of the same. However, this early detection is very difficult, since at the beginning of a crisis, symptoms can be very subtle, almost impossible to recognize, for both the patient and the family.

In bipolar disorder crises, common denominator is the change in sleep pattern and physical activity, which manifest weeks before a crisis. Early detection of these changes would allow more effective treatments, improve the ability of socio-labor integration of these patients and decrease the dose of pharmacological treatment necessary for his stabilization.

The Bip4Cast project developed between the Clinic of Nuestra Señora de La Paz (Orden Hospitalaria San Juan de Dios) and the GRASIA-GTeC group of the Complutense University of Madrid, seeks to predict by means of techniques such as actigraphy [4] (a medical technique that allows continuous monitoring sleep and physical activity) in patients with bipolar disorder. Besides possessing other data sources such as medical history of each patient, periodic evaluation of a doctor and mobile application that provides a questionnaire for doing a daily questionnaire and that in combination and process of all this information will implement an alarm system that notifies patient and his doctor the proximity of a crisis, allowing to take appropriate measures in order to avoid its appearance.

The project has been classified as Big Data mainly because of the volume of data it has to process, the variety of sources and other features that are detailed in section 4.2.1.

Defined the problem, will be designed as an initial step a Big Data solution to support the generation, acquisition, storage and analysis of data. In this work, the Big Data Working Group of the National Institute of Standards and Technology of the United States in English (NBD-PWG) [5] was used as reference. This implementation will be in a Hadoop [6] environment that is a platform designed for distributed data storage and processing, has horizontal scalability, is fault tolerant thanks to the HDFS distributed file system [7] that automatically replicates the data blocks to distinct nodes and is an open-source operating under the license of the Apache Software Foundation [6] [8].

There are a variety of Hadoop distributions that integrate tools that specialize in each layer of a Big Data architecture, as is the case of the Hortonworks distribution that is open source [10], unlike other distributions like Cloudera or MapR. With the use of a Hortonworks distribution it is possible to implement a Big Data architecture of lower cost and time. Considering these benefits, this has been chosen for the implementation of the proposed architecture.

1.1. Research Objective

The objective of the present work is to propose a Big Data architecture that considers the generation, acquisition, storage and analysis of data to support the prediction of crisis in patients diagnosed with bipolar disorder.

The proposed architecture proposes the use and appropriate choice of different technologies in the different levels of a Big Data structure according to the characteristics of the problem. In addition, it can process various sources of data among which we highlight: clinical history, periodic evaluations of the doctor, answers to daily questionnaires made to the patient from a mobile application about drug use, bedtime, etc; And mainly data related to physical activity and sleep captured in acrobat [4].

Likewise, it is proposed to perform tests of the operation of the architecture in the treatment of the main data source (accelerometer) since its collection to its processing.

1.2. Related work

The behaviors and patterns discovery from volumes of data from various sources either to obtain results in real time or for further analysis, has been applied in diverse areas of research and business such as health, social networking, security, marketing and finance, among other.

A Big Data architecture should be designed according to the needs of the problem or research to address where the type of data sources how structured, semi-structured, unstructured, the need for data processing: whether this will be in real time or no (batch processing), privacy: whether it is necessary to apply anonymization techniques or not, etc, are evaluated. In the article "Big Data Architecture for Pervasive Healthcare" [11] the main components of a Big Data architecture are established within the generalized healthcare that involves the collection of data through mobile devices and sensor networks that usually are large, diverse and have high frequency. The nature of Big Data, volume, variety and speed [12] coupled with its analytical capacity complement the provision of generalized healthcare. This study offers three important contributions: 1) identifies Big Data research themes and generalized health care, 2) establishes the relationship between research topics, which later become part of Big Data's architecture for generalized health care, and 3) proposes an overall Big Data architecture for generalized health care projects.

A Big Data solution can be implemented within an organization using its own infrastructure, but this entails a major investment that, in many cases, cannot be assumed. After evaluating the amount of data that will be collected, stored and processed, and estimating the amount of processing necessary to obtain the results, it will be possible to establish whether it is desirable to outsource all or part of the Big Data architecture. In the article "Big Data Storage Architecture Design in Cloud Computing" [13], it is proposed to solve the problem of traditional storage and data management of large volumes, with a platform built on Hadoop in a local infrastructure and another for storing data in the cloud. In order to break the bottlenecks of the master node, mainly related to the data update, is proposed a storage system of better performance hosted in the cloud that among other advantages how the recovery of failures that helps enormously in the management of the storage system (common bottleneck in updating data).

As described so far, there are a variety of architectural proposals for the implementation of Big Data solutions, so the need arises to define a standard for the components of the architecture and the operational models that together give rise to Called Big Data ecosystem. In the article "Defining architecture components of the Big Data Ecosystem" [14] the various domains of Big Data domain in science, industry and social activities are analyzed. The article proposes a definition of Big Data from its properties or characteristics, which include: volume, speed, variety, value and truthfulness; In addition to data models and structures, data analytics, infrastructure and

security. The proposed architecture (Big Data Architecture Framework) includes the following components: infrastructure, analytics, models and data structures, life cycle management and security.

The National Institute of Standards and Technology (NIST) [15] is an agency of the United States Department of Commerce that works to build consensus on Big Data's core concepts. This agency has created the NIST Big Data Public Working Group (NBD-PWG), a working group in charge of elaborating a series of volumes denominated NIST Big Data Interoperability Framework. In the present investigation, we have taken as reference the volume 6 [5], which summarizes the work done by the NBD-PWG to characterize Big Data from an architecture-based perspective, presents the conceptual model of a Big Data (NIST Big Data Reference Architecture - NBDRA) and the components of the NBDRA are analyzed. All this gives us a starting point to define our own architecture for the Bip4Cast project.

The use of accelerometers as sources of data have been several investigations that have made use of them. In the article "Actigraphy of the wrist" [4], the actigraphy is defined as a useful tool for the study of sleep, which allows its evaluation during prolonged periods of time. It is presented to the actigraphy as a valid estimate of TST (Total Sleep Time), percentage of sleep and WASO (Wake After Sleep Onset). Although this technique cannot replace other assessment tools, such as clinical interviews or sleep magazines, it can provide useful information in the evaluation of insomnia and circadian sleep rhythm disorders. The study of sleep is preponderant in the evaluation of the pattern of a patient diagnosed with bipolar disorder, therefore, the data obtained by this device are the main source within the proposed architecture.

In the article "A Novel, Open Access Method to Assess Sleep Duration Using a Wrist-Worn Accelerometer" [16], they use the data generated by accelerometers to elaborate an algorithm of general scope that evaluates the sleep of 60-83 participants who wear an accelerometer for 9 consecutive days. This algorithm states that a person is in a sleep period by identifying changes in arm angle greater than 5 degrees for 5 minutes or more, which is recorded as a participant's sleep period.

Detection of behavioral patterns to define states of sleep, sedentary lifestyle and activity were evaluated in the article "Behavioral Periodicity Detection from 24 h Wrist Accelerometry and Associations with Cardiometabolic Risk and Health-Related Quality of Life" [17] by observation of periodicities (repetition of patterns). Data generated by the accelerometer can

capture untapped patterns, including sleep, sedentary lifestyle and active behaviors. The accelerometer used for this study is the GENEActiv [18], the same used by the Bip4Cast project. The results show that by capturing data with this device it is possible to characterize patterns of behavior validated by other formal methods in the field of health.

The mentioned articles refer to the use of accelerometers for the capture of data, allowing to obtain patterns of sleep and physical activity. This reference has been considered as part of the data generation component within the proposed Big Data architecture.

1.3. Document structure

The present work is divided in 6 chapters:

Chapter 2 describes the approaches to the Big Data architectures and the concepts of the elements involved in each layer, as well as the use cases, Big Data requirements and the conceptual model of the architecture proposed by NIST Group.

Chapter 3 explores the Hadoop ecosystem and the tools that it offers for implementing a Big Data environment, from data ingestion to analysis and visualization of results. It also performs an analysis of the most important features in the main Hadoop distributions and a comparison between them.

Chapter 4 details the Big Data architecture proposed by this work for the development of the Bip4Cast project. Explain the details of an architecture that addresses the generation, collection, storage and analysis of the necessary data, defining the appropriate technologies and tools according to the characteristics of the project.

Chapter 5 details the implementations and configurations made of the various technologies defined in the architecture model proposed for the project.

Chapter 6 presents the tests and experiments performed from data capture with the SFTP service to processing within the Hadoop Hortonworks distribution to verify the operation of the proposed architecture.

Finally, chapter 7 presents the conclusions and future lines of work for the project.

Capítulo 2. Arquitecturas Big Data

En el presente capítulo se va a detallar las diferentes arquitecturas utilizadas en un ecosistema Big Data y las capas más importantes como seguridad, gestión, generación, adquisición, almacenamiento y análisis, los actores, tecnologías y herramientas que forman parte de la arquitectura. Estas tecnologías y herramientas son implementadas según las características del proyecto o tipo de investigación a realizar, es por eso que se van a definir los componentes funcionales de una arquitectura Big Data, resaltando en qué casos suelen ser más útiles o cómo en combinación con otras pueden aportar mejores resultados [19].

Si hablamos de Big Data, esta no es una sola tecnología, sino una combinación de viejas y nuevas tecnologías que se integran para poder abordar las nuevas características de los datos como velocidad, variedad y volumen. Por lo tanto, Big Data es la capacidad de manejar un gran volumen de datos de diversas fuentes, a la velocidad correcta, y dentro del marco de tiempo adecuado para permitir el análisis ya sea posterior a la recolección de los datos o en tiempo real [20]. Big Data es típicamente dividido en tres características que son las 3Vs [21]. El volumen que es la cantidad de datos, la velocidad que hace referencia la tasa de flujo de los datos en la creación, almacenamiento, análisis y visualización, y variedad que son las distintas fuentes de datos. Aunque se tiende a simplificar Big Data en 3Vs existen propuestas que hacen referencia a otras como la *variabilidad* que se refiere a cualquier cambio de los datos en el tiempo como puede ser la tasa de transferencia o el formato [22], la *veracidad* la cual indica la exactitud o precisión de los datos [14]. Por lo que no debe entenderse la definición de Big Data limitada a solo 3Vs; por ejemplo, puede darse el caso de una cantidad relativamente pequeña de datos muy diversos y complejos o es posible que se procese un gran volumen de datos muy simples. Esos datos simples pueden ser estructurados, semiestructurados o no estructurados. Es por eso que se suele incluir la V de *valor* que hace referencia al aporte de valor a la organización de parte del análisis de los datos través del procesamiento Big Data [23][22]. Esto nos indica, por ejemplo, cuan precisos son los datos elegidos para predecir el valor del negocio o si en realidad tiene sentido los resultados del análisis de Big Data.

2.1. Casos de uso y requisitos de Big Data

Antes de profundizar en la arquitectura, es importante tener en cuenta los casos de uso de Big Data y los requerimientos o requisitos dentro de su arquitectura. El grupo de trabajo del NIST [24] proporciona información de varios ejemplos de arquitecturas de Big Data, incluyendo áreas como la salud y el gobierno. En la Tabla 2.1 se ilustran los 9 dominios de aplicación de Big Data que están definidos.

Tabla 2.1. Dominios de aplicación de Big Data

Dominios
Gobierno
Comercio
Defensa
Salud y Ciencias de la vida
Deep Learning y Social Media
Ecosistemas de Investigación
Astronomía y Física
Tierra y Medio ambiente
Energía

De los dominios definidos a partir de los casos de uso que fueron analizados, se han extraído 7 categorías genéricas a partir de los requisitos de cada uno de estos, los cuales se muestran en la Tabla 2.2 donde se hace referencia a la relación entre las categorías definidas y los componentes de una arquitectura Big Data. A continuación, se detallan dichos requisitos organizados en categorías.

Requisitos de Fuentes de Datos (DSR por sus siglas en inglés)

- DSR-1: Soporte para procesamiento en tiempo real, asincrónico, streaming o por procesamiento por lotes para recopilar datos de fuentes centralizadas, distribuidas, en la nube, de sensores o instrumentos.
- DSR-2: Soporte para la transmisión de datos a bajas frecuencias, a ráfagas o a una alta tasa de transmisión de datos entre las fuentes y los clústeres de computación.
- DSR-3: Soporte para una diversa variedad de fuentes que van desde datos estructurados, semiestructurados y no estructurados los cuales pueden ser: textos, documentos, gráficos, web, geoespaciales, comprimidos, temporizados, espaciales, multimedia, simulaciones e instrumentales.

Requisitos del Proveedor de Transformación (TPR por sus siglas en inglés)

- TPR-1: Soporte a diversas técnicas de cálculo intensivo, procesamiento estadístico y análisis gráfico y machine learning.
- TPR-2: Soporte al procesamiento analítico por lotes o en tiempo real.
- TPR-3: Soporte al procesamiento a una gran variedad de tipos de datos.
- TPR-4: Soporte al procesamiento de datos en movimiento (streaming, búsqueda de nuevos contenidos, rastreo, sistemas de recomendación, etc.)

Requisitos del Proveedor de Capacidades (CPR por sus siglas en inglés)

- CPR-1: Soporte a paquetes de software antiguos y avanzados.
- CPR-2: Soporte a plataformas de computación distribuidas en clusters, coprocesadores, procesamiento de entrada/salida (infraestructura).
- CPR-3: Soporte para transmisión de datos de manera flexible o elástica (redes de datos).
- CPR-4: Soporte para almacenamiento de datos distribuido, de grandes volúmenes y avanzado (almacenamiento).

Requisitos del Consumidor de Datos (DCR por sus siglas en inglés)

- DCR-1: Soporte para búsquedas de alta velocidad desde datos procesados con alta relevancia y exactitud.
- DCR-2: Soporte a una diversidad de formatos de archivos para visualización, representación y reportes.
- DCR-3: Soporte de diseño visual para la presentación de resultados.
- DCR-4: Soporte para interfaces de usuario mediante el uso del navegador o herramientas de visualización.
- DCR-5: Soporte para la visualización de datos en alta resolución o en capas multidimensionales.
- DCR-6: Soporte para resultados en streaming hacia los usuarios.

Requisitos de Privacidad y Seguridad (SPR por sus siglas en inglés)

- SPR-1: Soporte para preservar y proteger la seguridad y privacidad de los datos sensibles.
- SPR-2: Soporte para la autenticación basada en políticas de sistemas de pruebas aislados (sandbox), control de acceso y multiniveles en los datos protegidos.

Requisitos de Gestión

- LMR-1: Soporte para la calidad de datos incluyendo pre-procesamiento, data clustering, clasificación, reducción y transformación de formato.
- LMR-2: Soporte para actualizaciones dinámicas de los datos, perfiles de usuario y enlaces.
- LMR-3: Soporte al ciclo de vida de los datos y políticas de preservación a largo plazo incluyendo el origen de los datos.
- LMR-4: Soporte para la validación de datos.
- LMR-5: Soporte para la anotación o remarcar de parte del usuario en la validación de los datos.
- LMR-6: Soporte para la prevención de pérdida de los datos o la corrupción de los mismos.
- LMR-7: Soporte para tener archivos en múltiples sitios.
- LMR-8: Soporte para un identificador persistente de datos y el seguimiento de los mismos.
- LMR-9: Soporte para la estandarización, agregación y normalización de datos que provienen de diferentes fuentes.

Otros Requisitos (OR)

- OR-1: Soporte a interfaces de usuario desde plataformas móviles para acceder a los resultados.
- OR-2: Soporte a la monitorización del procesamiento analítico desde plataformas móviles.
- OR-3: Soporte para la visualización de contenido de búsquedas y su representación en plataformas móviles.
- OR-4: Soporte para la adquisición de datos del dispositivo móvil.
- OR-5: Soporte de seguridad a través de los dispositivos móviles.

Los requisitos descritos líneas arriba, no engloban necesariamente todos los casos de uso de las soluciones Big Data, estos deben ser asumidos como una referencia al definir una arquitectura ya que se puede tener características especiales y por lo tanto otros requisitos no especificados. El desarrollo de una arquitectura de referencia de Big Data requiere una continua actualización frente a la aparición de nuevas tecnologías en la generación, captura, tratamiento y análisis de los datos y un alto conocimiento de las técnicas actuales, problemas e inquietudes. Es así, que el grupo NBD-PWG del NIST [5] ha evaluado diversos casos de uso para obtener una

comprensión adecuada de las aplicaciones actuales de Big Data, con lo que se puede identificar aspectos comunes dentro de las arquitecturas de Big Data.

Tabla 2.2. Referencia de los requisitos de Big Data con los componentes de la arquitectura

Categorías de requisitos		Referencia a los componentes de la arquitectura
Fuentes de datos	→	**Proveedor de datos**
Transformación de datos	→	**Proveedor de aplicaciones Big Data**
Capacidades	→	**Proveedor de infraestructura Big Data**
Consumo de datos	→	**Consumidor de datos**
Privacidad y seguridad	→	**Estructura de Privacidad y Seguridad**
Gestión del ciclo de vida	→	**Sistema de Orquestación; Estructura de Gestión**
Otros requerimientos	→	**Para todos los componentes y estructuras**

2.2. Actores principales en una arquitectura

Dentro de una arquitectura Big Data existen diversos actores que cumplen roles de gestión, operación, procesamiento, seguimiento y seguridad, entre otros, además de interactuar entre ellos o con otros componentes externos e internos de la arquitectura. En la Tabla 2.3 se hace referencia a los principales y potenciales actores a partir de los 7 componentes definidos en la Tabla 2.2 los cuales vienen a ser los roles generales dentro la arquitectura.

Tabla 2.3. Principales actores dentro de una arquitectura Big Data [5]

Rol	Actor
Sistema de Orquestación	Líder de la Empresa, Consultores, Científicos de Datos, Arquitectos de la Información, Arquitectos de Software, Arquitectos de Seguridad, Arquitectos de Privacidad, Arquitectos de Redes de Datos
Proveedor de Datos	Empresas / Organizaciones, Agencias Públicas, Científicos e Investigadores, Motores de búsqueda, Aplicaciones Web, FTP y otros; Operadores de Red, Usuarios finales
Estructura de Seguridad y Privacidad	Oficial / Jefe de seguridad, Especialista en seguridad
Proveedor de aplicaciones Big Data	Especialista en Aplicaciones, Especialista en Plataformas, Consultores
Proveedor de infraestructura Big Data	Clústeres In-house, Centros de Datos, Proveedores Cloud
Consumidor de Datos	Usuarios finales, Investigadores, Aplicaciones, Sistemas
Estructura de Gestión	Personal In-house, Gestor del Centro de Datos, Proveedores cloud

2.3. Modelo conceptual de una arquitectura Big Data

El grupo NBD-PWG del NIST [5] ha definido un modelo conceptual de una arquitectura Big Data que considera cinco componentes funcionales conectados por interfaces de interoperabilidad, dos estructuras que envuelven a todos los componentes, lo que muestra la interrelación de la gestión, seguridad y privacidad con dichos componentes como se muestra en la Figura 2.1.

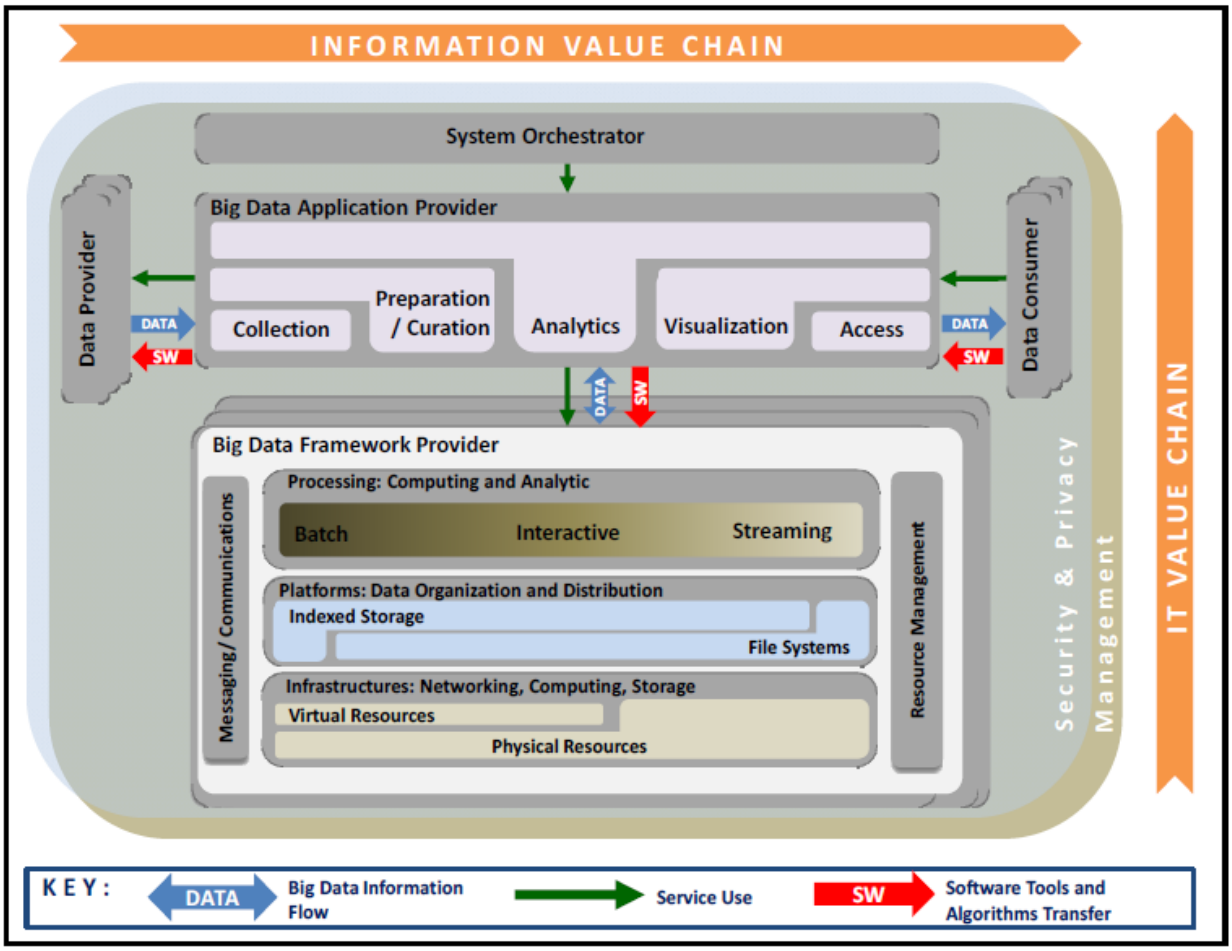


Figura 2.1. Modelo Conceptual de la arquitectura de referencia del NIST [5]

2.3.1. Componentes funcionales de la arquitectura

La arquitectura se organiza en torno a dos ejes que representan las dos cadenas de valor de Big Data (Figura 2.1). En el eje horizontal se tiene la Información y en el eje vertical las Tecnologías de la Información (TI). En cuanto al eje de la Información, el valor se crea mediante la recolección de datos, integración, análisis y la aplicación de los resultados siguiendo la cadena de valor [25]. Por otro lado, en el eje de las TI el valor se crea brindando redes de datos, infraestructura,

plataformas, herramientas de aplicación y otros servicios TI para el alojamiento y operación de implementaciones Big Data. Uno de los componentes que proporciona valor a ambos ejes, según sus necesidades, es el proveedor de aplicaciones Big Data que por medio del análisis de datos proporciona valor para las partes interesadas (*stakeholders*) de ambas cadenas de valor. A continuación, se pasa a detallar las características principales de cada componente.

A. Sistema de orquestación

Proporciona los requisitos generales que debe cumplir el sistema, como son la gobernabilidad, la arquitectura, los recursos y los requerimientos del negocio. También se incluyen las tareas de supervisión o auditoría para asegurar que el sistema cumple con dichos requisitos.

El rol de orquestación proporciona los requisitos del sistema y la monitorización del sistema de datos. Este implica una colección de funciones más específicas, que pueden ser ejecutadas por uno o más actores, que gestionan y orquestan la operación del sistema Big Data. Estos actores pueden ser usuarios, componentes de software o alguna combinación de ambos. Las cargas de trabajo administradas por este componente pueden ser asignadas a nodos físicos o virtuales a un bajo nivel o proporcionar una interfaz gráfica de usuario donde se pueda realizar la especificación de los flujos de trabajo y que además integre múltiples aplicaciones y componentes de alto nivel.

B. Proveedor de datos

Este rol hace que los datos estén disponibles para sí mismo como para otros roles. Para cumplir con su función, crea una abstracción de diversos tipos de fuentes de datos (como datos en bruto o datos previamente transformadas por otro sistema) y los pone a disposición a través de diferentes interfaces funcionales. El actor que desempeña este rol puede formar parte del sistema Big Data de manera interna o externa a la organización y pueden ser desde un sensor, un usuario que introduce datos de forma manual, a otro sistema Big Data.

Las fuentes de datos pueden ser registros internos o públicos, audio, imágenes, vídeos, datos de sensores, web logs, registros de auditoría, cookies HTTP, entre otras fuentes.

El proveedor de datos incluye actividades que son comunes a la mayoría de sistemas de gestión de datos:

- recolección de datos;
- persistencia de los datos;

- proporcionar funciones de transformación para la depuración de datos confidenciales o sensibles;
- creación de metadatos que describen las fuentes de datos, políticas de acceso y otros atributos relevantes;
- cumplimiento de los derechos asignados en el acceso a los datos;
- establecimiento de contratos formales e informales para la autorización de acceso a datos;
- hacer que los datos sean accesibles a través de interfaces intuitivas donde la información sea requerida o solo enviada por el sistema sin necesidad de hacer una solicitud (paradigma push/pull [26]);
- publicar la disponibilidad de la información y los medios para acceder a ella.

C. Proveedor de aplicaciones Big Data

El proveedor de aplicaciones Big Data lleva a cabo un conjunto de operaciones en el ciclo de vida de los datos para cumplir con los requisitos establecidos por el Sistema de Orquestación. Es aquí donde se combinan las capacidades generales de la plataforma Big Data para producir un sistema de datos específico. Aquí es donde desarrollamos la funcionalidad y la lógica del negocio que será ejecutada por el sistema Big Data. El proveedor de aplicaciones de Big Data puede ser una sola instancia o una colección de estas, que se ejecutan en las diferentes etapas del ciclo de vida de los datos. Este rol consta de actividades que son representadas como subcomponentes (Figura 2.1), las cuales se detallan a continuación.

- **Recolección de datos**

  La actividad de recolección de datos está integrada con el componente proveedor de datos. Esta puede ser un servicio como un servidor de archivos o un servidor web configurado por el sistema de orquestación para realizar recolecciones particulares de datos, también puede ser una aplicación diseñada para extraer o recibir datos desde el proveedor de datos. La persistencia de los datos puede no ser necesaria debido que se pueden utilizar procesamientos en memoria u otros servicios proporcionados por el proveedor de infraestructura Big Data.

- **Preparación**

  En la actividad de preparación es donde se podría indicar que se lleva a cabo el proceso de transformación de un ciclo ETL (Extract, Transform, Load) [27], aunque la actividad de

análisis también puede ser considerada como la ejecución de tareas avanzadas de la transformación. Entre las tareas que realiza esta actividad se pueden incluir a la validación de datos, limpieza, normalización, entre otras.

- **Análisis**

  La actividad implementa los métodos y técnicas para extraer conocimiento de los datos basados en los requisitos del sistema. Por lo general, esta actividad proporciona software para el análisis en *streaming* o por procesamiento por lotes. La plataforma de comunicación y mensajería del proveedor de infraestructura Big Data puede ser utilizado para transferir datos o funciones de control a las aplicaciones que se ejecutan en las plataformas de procesamiento.

- **Visualización**

  La actividad de visualización prepara los elementos de los datos procesados y los resultados de la actividad de análisis para su presentación al consumidor de datos. El objetivo de esta actividad presentar los datos con un formato que permita expresar de manera óptima su significado y el conocimiento que aporta. Los formatos de presentación pueden ser informes basados en texto o mostrar los resultados del análisis en forma gráfica [28]. Esta actividad interactúa con la actividad de acceso, la actividad de análisis y con el proveedor de infraestructura Big Data para que se pueda ofrecer una visualización interactiva al consumidor de datos. La visualización puede ser una implementación de una aplicación, la integración de una o más librerías o se puede utilizar plataformas especializadas en el procesamiento de visualización.

- **Acceso**

  La actividad de acceso se centra en la comunicación e interacción con el consumidor de datos. Del mismo modo que la actividad de recolección, esta puede ser un servicio genérico, como un servidor web o un servidor de aplicaciones configurado para gestionar peticiones específicas del consumidor de datos. Esta actividad se comporta como una interfaz entre las actividades de visualización y análisis para responder a las solicitudes del consumidor de datos y utiliza las plataformas de procesamiento e infraestructura para recuperar datos y poder atender a las solicitudes del consumidor de datos. La interfaz con el consumidor de datos puede ser síncrona o asíncrona y puede utilizar el paradigma *push/pull* [26] para la transferencia de datos.

D. Proveedor de infraestructura Big Data

Este rol tiene los recursos o servicios generales para ser utilizados por el proveedor de aplicaciones Big Data en la creación de una aplicación específica. Existen una variedad de componentes como recursos de procesamiento, almacenamiento y redes de datos de donde el proveedor de aplicaciones Big Data puede elegir para construir un sistema específico. El proveedor de infraestructura Big Data consiste en una o más instancias de tres subcomponentes: plataforma de infraestructura, plataforma de datos y plataforma de procesamiento, los que detallamos a continuación.

- **Infraestructura**

  Este elemento proporciona todos los recursos necesarios para albergar y ejecutar las actividades de los otros componentes. Una clasificación general de estos recursos se da de la siguiente manera:

  - **Redes:** Estos son los recursos que transfieren datos de un componente de la infraestructura a otro.
  - **Computación:** Son los procesadores y la memoria física que ejecutan y mantienen el software de los demás componentes.
  - **Almacenamiento:** Proporcionan la persistencia de los datos en un sistema Big Data.
  - **Estructura o Ambiente:** Son los recursos de la estructura física como la energía, refrigeración o seguridad que deben tenerse en cuenta cuando se realiza una implementación de una plataforma Big Data, lo cual también podría delegarse si la implementación se realiza sobre servicios de terceros alojados en la nube lo que se conoce como una infraestructura como servicio (IaaS por sus siglas en inglés).

- **Plataforma de datos**

  Permite la organización y distribución lógica de los datos en combinación con los accesos asociados entre las *API (Application Programming Interface)* y los métodos. La organización lógica de los datos puede variar desde archivos planos delimitados hasta almacenes de datos relacionales o por columna en un entorno totalmente distribuido. Los medios de almacenamiento van desde cintas de almacenamiento, medios magnéticos, discos de estado sólido hasta las memorias de acceso aleatorio. Por estas características de almacenamiento, los métodos de acceso pueden variar desde *API* de acceso a archivos hasta lenguajes como el *SQL (Structured Query Language).*

- **Plataforma de procesamiento**

  Proporcionan el software necesario para dar soporte a la implementación de aplicaciones que cumplen con las características de Big Data. Esta plataforma se centra en dar soporte a la manipulación de datos que puede ser por procesamiento por lotes *(batch processing)* o por *streaming*. Cuenta con tres fases de procesamiento: la ingesta de datos, el análisis de datos y la difusión de datos, los cuales acompañan al flujo de datos a través de la arquitectura. En el procesamiento por lotes, por *streaming* o la combinación de ambos se puede aplicar a las tres fases antes mencionadas.

  Muchos algoritmos y modelos de procesamiento se han definido con la evolución del tratamiento de datos de los que se puede resaltar dos de los más conocidos en el espacio de Big Data, *MapReduce* [29] y *Bulk Synchronous Parallel (BSP)* [30]. Siendo la principal diferencia de que *BSP* puede realizar cambios en los datos que se procesan. Se deben considerar las ventajas y desventajas de ambos al momento de su implementación, por ejemplo, *BSP* cuenta como desventaja el alto costo en la sincronización; mientras que *MapReduce* no tiene un buen funcionamiento cuando necesita el acceso iterativo a partes del conjunto de datos ya que se tendría que volver a leer todo el conjunto de datos.

E. Consumidor de datos

El consumidor de datos recibe el valor de salida del sistema Big Data. Después de que el sistema añade valor a las fuentes de datos originales, el proveedor de aplicaciones Big Data entrega ese mismo tipo de interfaces funcionales hacia el consumidor de datos. Las actividades asociadas con el rol del consumidor de datos son los siguientes:

- búsqueda y recuperación,
- descarga,
- análisis local,
- informes,
- visualización, y
- datos que se utilizan para sus propios procesos.

El consumidor de datos utiliza las interfaces o servicios proporcionados por el proveedor de aplicaciones Big Data para obtener acceso a la información requerida. Estas interfaces pueden incluir la presentación de datos, recuperación de datos y la representación de datos.

F. Capa de seguridad y privacidad

Los temas de seguridad y privacidad afectan a todos los otros componentes de una arquitectura Big Data. Este rol interactúa con el sistema de orquestación en cuanto a políticas, requisitos y auditoría; también con el proveedor de aplicaciones Big Data y el proveedor de datos Big Data en cuanto al desarrollo, implementación y operación de estos. Por ejemplo, en el caso de una compañía de salud, es probable que se desee utilizar aplicaciones de Big Data para determinar los cambios en la demografía o los cambios en las necesidades del paciente. Estos datos acerca de los pacientes necesitan ser protegidos tanto para satisfacer los requisitos establecidos y proteger la privacidad de los pacientes. Por lo que, una función clave es la de definir lo niveles de accesos para quienes puedan ver los datos y en qué circunstancias se les permite hacerlo. Además, de verificar la identidad de los usuarios del sistema, así como proteger la identidad de los pacientes. Estos tipos de requisitos de seguridad tienen que ser parte de la gran estructura de datos.

G. Capa de gestión

Las características de Big Data [22] demandan un sistema versátil, una plataforma de gestión de software, junto con la gestión y monitorización de los recursos y su rendimiento. La gestión de Big Data implica el sistema, los datos, la seguridad y las consideraciones de privacidad, mientras se mantiene una alta calidad de los datos y una accesibilidad segura. Esta capa de gestión abarca dos grupos generales de actividades: gestión de sistemas y gestión del ciclo de vida de Big Data. El sistema de gestión incluye actividades como el aprovisionamiento, la configuración, la gestión de paquetes, gestión de software, gestión de copias de seguridad y la gestión de recursos. La gestión del ciclo de vida de Big Data implica actividades en torno al ciclo de vida de los datos como los de recolección, preparación / curación, análisis, visualización y acceso.

Para que los roles puedan cumplir un funcionamiento óptimo dentro de una arquitectura Big Data se debe prestar especial atención a la validación de los datos. Si para el proyecto se van a tener una combinación de fuentes de datos, es fundamental el poder validar que estas fuentes tienen sentido por sí mismas o cuando se combinan. Además, ciertas fuentes de datos, sobre todo los datos médicos, pueden contener información sensible por lo que se debe poner en práctica un nivel suficiente de seguridad y gobernabilidad. Por supuesto, el diseño de una arquitectura Big

Data en primer lugar tiene que comenzar con la definición del problema para luego establecerse ciertas características como el tipo de datos o combinación de varios de ellos, si es necesario el procesamiento en tiempo real o un procesamiento por lotes, por mencionar algunas consideraciones a tomar en cuenta.

2.4. Otros modelos de referencia

Durante la investigación se han encontrado algunas propuestas de referencia de una arquitectura Big Data como la que se muestra en el libro *"Big Data For Dummies"* [27] que presenta la composición de una arquitectura con los siguientes componentes (ver Figura 2.2): Interfaces y fuentes de datos, Infraestructura física, Infraestructura de seguridad, Fuentes de datos operacionales, Organización de servicios y herramientas y Almacenes de Análisis de Datos los cuales pasamos a detallar a continuación.

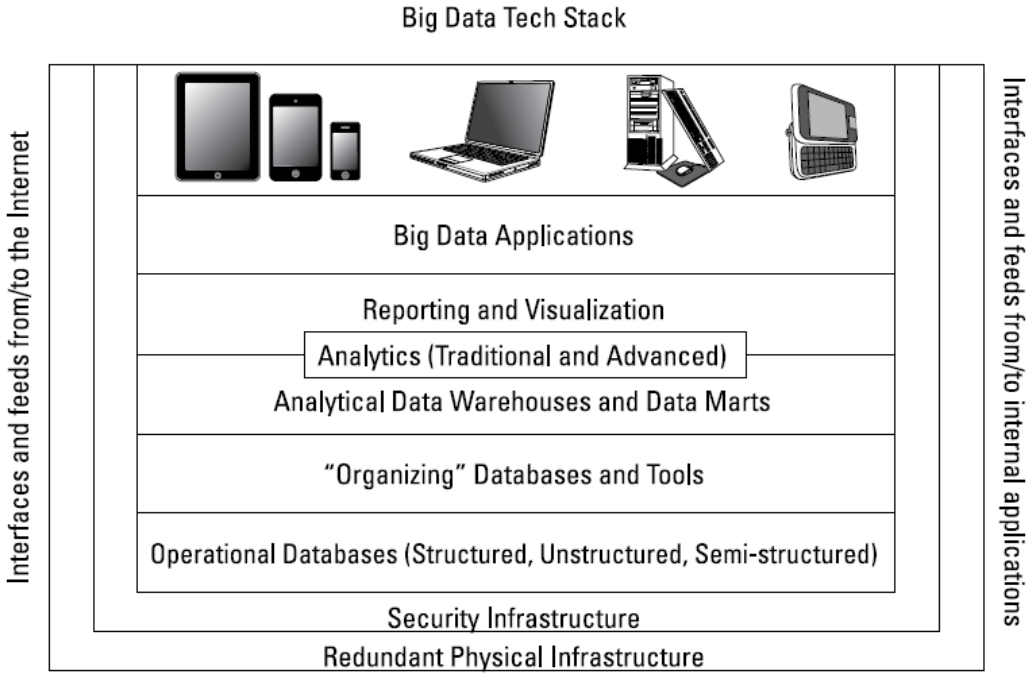


Figura 2.2. Referencia de arquitectura Big Data

- **Interfaces y fuentes de datos**

Se hace referencia a las interfaces y las fuentes de datos internos y externos. Dado que, la característica que hace a Big Data tener un continuo crecimiento en el volumen es la de recoger una gran cantidad de datos de muchas fuentes. Por lo tanto, el desarrollo y uso de diversas interfaces como las interfaces de programación de aplicaciones (API) se convierten en un punto

clave para cualquier arquitectura Big Data. Además, se debe tener en cuenta que en todos los niveles y capas de una arquitectura existen interfaces especializadas para cada una de estas.

- **Infraestructura física**

Al igual que en la propuesta del NIST esta capa brinda soporte a la infraestructura física y virtual de una arquitectura Big Data. En esta propuesta se resalta a la redundancia debido a que al existir un manejo de muchos datos de fuentes diferentes se hace necesario contar con mecanismos de replicación tanto a nivel hardware como de software. Además, existen servicios externos en la nube que nos permiten incrementar los recursos internos. Uno de estos servicios es la plataforma de Software como Servicio (SaaS) que permite hacer el análisis de datos como un servicio.

- **Infraestructura de seguridad**

En esta capa se hace referencia a las políticas de seguridad, niveles de acceso, roles y otras implementaciones de seguridad dentro de una arquitectura Big Data para garantizar la validez de la información a ser obtenida. Es por eso, que el asegurar los datos como los procesos de análisis se convierte en una tarea importante dentro de una arquitectura Big Data.

- **Fuentes de datos operacionales**

En esta capa de la arquitectura se gestiona fuentes de datos operativas como una base de datos relacional, fuentes no estructuradas, tales como datos de clientes o de las redes sociales por citar algunos ejemplos. Por tal motivo es que una arquitectura Big Data cuenta con modelos de bases de datos de documentos, grafos, columnas y geoespaciales [31], la los que se denominan como base de datos NoSQL *(Not only SQL)*.

- **Organización de servicios y herramientas**

Este componente permite realizar la organización de los servicios y las herramientas para la captura, validación e integración de varios elementos Big Data relevantes. La función de organización de consiste en un ecosistema de herramientas y tecnologías que se pueden utilizar para recoger y reunir datos para su posterior procesamiento. De modo que, las herramientas deben contar con características como integración o escalabilidad.

- **Almacenes de datos para analítica**

Esta capa se encarga de almacenar el subconjunto de datos ya procesados de donde se obtendrán los patrones y resultados, y estarán disponibles para el proyecto o negocio. La

actualización de estos almacenes se realiza a menudo por medio de procesamiento por lotes, lo cual puede no ser óptimo para algunas arquitecturas donde se prioriza el procesamiento por *streaming.*

En el artículo *"Big Data Storage Architecture Design in Cloud Computing"* [13] se muestra un modelo de arquitectura general para Big Data que tiene dos sistemas estándar, el sistema estándar de seguridad y el sistema estándar de gestión de servicios, una capa de interfaces (front layer), capa núcleo (core layer), capa de gestión, capa de datos y la capa de aplicación. Cabe resaltar que esta propuesta implementa el servicio de almacenamiento en la nube. Ver Figura 2.3.

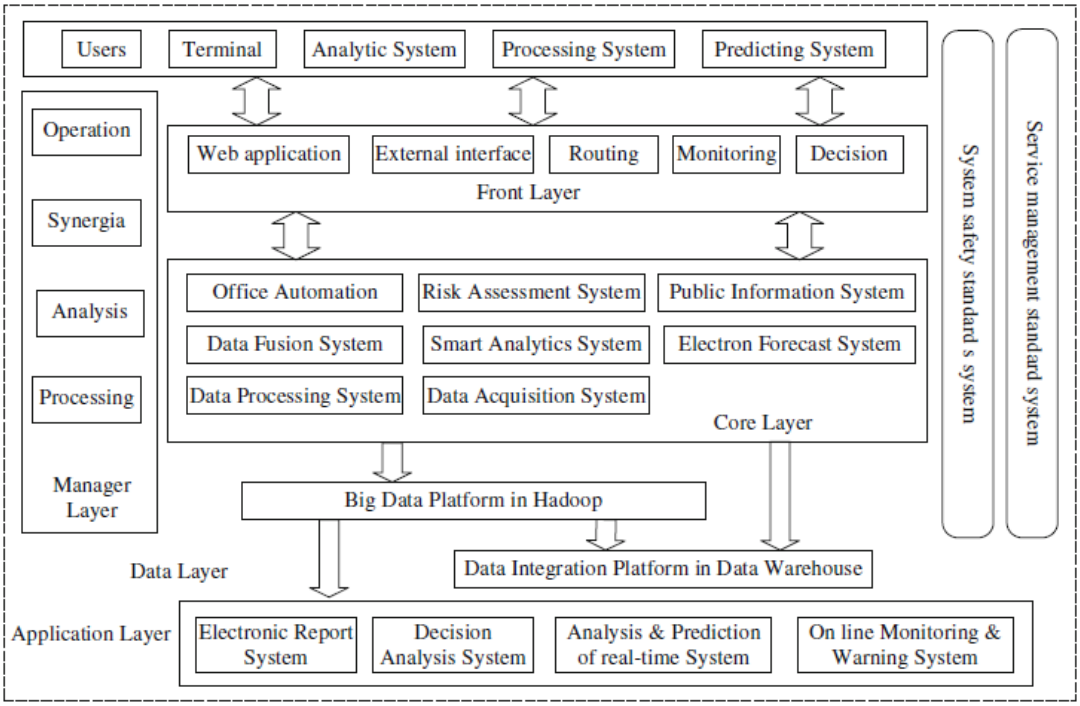


Figura 2.3. Modelo general de una arquitectura Big Data

En el artículo *"Toward Scalable Systems for Big Data Analytics: A Technology Tutorial"* [32] se presenta las capas que forman parte de una arquitectura Big Data, pero no un modelo de referencia como tal para poder implementar una arquitectura propia. Este artículo se basa en la descripción de las tecnologías que pueden ser usadas en la definición de implementación de la arquitectura.

Otra propuesta de plataforma y arquitectura Big Data plantea cinco componentes los cuales son: infraestructura Big Data, analítica Big Data, estructuras y modelos de datos, gestión del ciclo de vida Big Data y seguridad Big Data [14], de esta propuesta se puede observar que no existe

diferencia sustancial con la presentada por el grupo NBD-PWG del NIST, dado que también se basa en el estudio realizado por el NIST.

Durante la investigación no se han encontrado mayores referencias de modelos conceptuales de arquitecturas Big Data, dado que en la mayoría de los casos los estudios se centran en alguna de las tecnologías, componentes de la arquitectura (por ejemplo, analítica de datos o Machine Learning) o alguna solución que refleja sólo una pequeña parte del problema. Es por eso que se ha elegido la propuesta del grupo de trabajo NBD-PWG del NIST [5] como base para la implementación de la arquitectura.

2.5. La arquitectura Lambda

En las secciones anteriores se ha descrito las capas más comunes de una arquitectura Big Data. Estas capas pueden, según el tipo de dato o necesidad de resultados específicos, requerir procesamiento por lotes, en tiempo real *(streaming)* o la combinación de ambos. Cabe resaltar que el procesamiento por lotes es todavía una práctica común en la arquitectura de datos de la mayoría de las organizaciones, y que el procesamiento en *streaming* no justifica una necesidad real cuando se puede transmitir los datos y almacenarlos en un sistema de archivos para luego procesarlos. La implementación de una arquitectura de *streaming* puede ser compleja cuando se hace la migración desde una plataforma orientada al procesamiento por lotes. Por ello, una arquitectura en streaming trae mucha más complejidad en términos de operaciones porque los componentes que dan soporte a esta siempre deben estar activos y debe existir un nivel de tolerancia a fallos muy alto; esta solución puede absorber los recursos disponibles cuando se producen requerimientos inesperados de datos.

La arquitectura Lambda toma lo mejor del procesamiento por lotes y por *streaming*. Los datos son transportados y procesados en tiempo real en una capa de velocidad, re-calcula y se utiliza para crear agregados parciales en la capa de carga, y finalmente se fusiona y se sirve en la capa de servicio. Ver Figura 2.4.

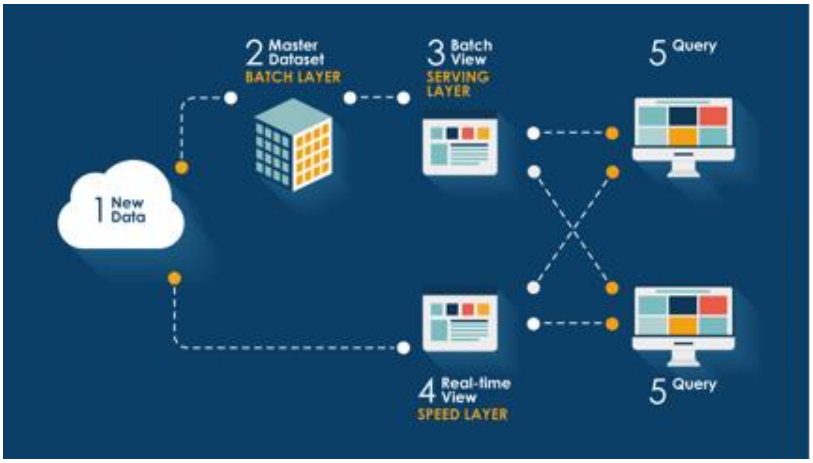


Figura 2.4. Arquitectura Lambda [33]

2.6. La arquitectura Kappa

La arquitectura Kappa es una simplificación de la arquitectura Lambda. Como se indica en *www.kappa-architecture.com*, Kappa es una arquitectura Lambda con el sistema de procesamiento por lotes eliminado. En lugar de utilizar una base de datos relacional SQL o un almacén de clave-valor como Cassandra, el almacén de datos en una plataforma Kappa es un registro de solo escritura. Desde el registro, se envían los datos a través de un sistema computacional y son puestos en los almacenes auxiliares de la capa de servicio. Para reemplazar el procesamiento por lotes, los datos simplemente son alimentados a través del sistema de transmisión. Ver figura 2.5.

Figura 2.5. Arquitectura Kappa [33]

En este capítulo se ha definido el modelo conceptual de una arquitectura Big Data, indicándose los diversos elementos o capas que la conforman, además de especificar el comportamiento y la relación de cada uno de ellos. Esto nos permitirá definir una arquitectura teniendo en consideración las diferentes características de cada elemento y cómo se deben organizar en una arquitectura específica. Y de esta manera, poder seleccionar y clasificar las herramientas que permitirán la implementación del sistema Big Data.

Capítulo 3. El Ecosistema Hadoop

En este capítulo se describirán las características principales del ecosistema *Hadoop,* así como las principales aplicaciones o herramientas que han sido establecidas según los roles de la arquitectura desarrollada en el Capítulo 2. Abordaremos las principales soluciones que integran estas herramientas basadas en Hadoop, y veremos cómo nos permiten implementar una arquitectura desde un nivel de abstracción alto para poder llevar a cabo diversos proyectos dentro del ecosistema Big Data.

Una vez definidas las herramientas que conforman este ecosistema y las distribuciones que las integran, llevaremos a cabo la selección e implementación dentro del modelo de arquitectura para el proyecto Bip4Cast.

3.1. La plataforma Hadoop

Es una plataforma de código abierto escalable y tolerante a fallos que permite el almacenamiento y procesamiento distribuido de grandes conjuntos de datos en *hardware* genérico [6]. Fue desarrollado por la Fundación Apache Software y utiliza el modelo de programación llamado MapReduce [29].

3.1.1. Características de Hadoop

Como se menciona en la definición de *Hadoop,* este cuenta con características que nos permiten utilizarlo como base en una implementación Big Data. A continuación, se detallan dichas características:

A. Escalabilidad

Un clúster *Hadoop* puede estar compuesto desde una sola máquina hasta, literalmente, miles de máquinas.

B. La tolerancia a fallos

Los servicios de *Hadoop* llegan a ser tolerantes a fallos gracias a la redundancia. Por ejemplo, en el caso del sistema de archivos distribuido de Hadoop *(Hadoop Distributed File System - HDFS),* considerando que el clúster tiene al menos tres nodos (por defecto), se replican automáticamente los bloques de datos a esos tres nodos separados.

C. Código abierto

El desarrollo de *Hadoop* está a cargo de la comunidad gobernada bajo la licencia de la *Apache Software Foundation*. Aquí se puede aportar en la mejora de *Hadoop* mediante la adición de características, corrección de fallos de software, mejora del rendimiento o en la escalabilidad.

D. Almacenamiento y procesamiento distribuido

Los grandes conjuntos de datos se dividen automáticamente en trozos más pequeños, llamados bloques, que se distribuyen a través de los nodos del clúster. Cada máquina procesa su bloque local de datos, lo que significa que el procesamiento se distribuye, a la vez, mediante cientos de *CPU* y *gigabytes* de memoria.

E. Hardware genérico

Todos los servicios de *Hadoop* se pueden ejecutar en *hardware* genérico *(commodity hardware)*. Con esto no solo reduce los costos de implementación, sino también, los costos de soporte y mantenimiento.

De manera más específica, diríamos que Hadoop es un motor que provee almacenamiento, vía HDFS; y cómputo a través de las capacidades de YARN *(Yet Another Resource Negotiator).*

3.1.2. Servicios y herramientas del ecosistema Hadoop

Existen diversas plataformas y herramientas que interactúan dentro del ecosistema *Hadoop*. En esta sección se describirán las más comunes y las que se usarán en la implementación de la arquitectura del proyecto Bip4Cast, agrupados según los componentes de la arquitectura planteada por el grupo de trabajo del NIST, descrito en el Capítulo 2. Cabe resaltar, que algunas de estas plataformas pueden abarcar más de un rol dentro de la arquitectura.

A. Aplicaciones para el componente de orquestación del sistema

En este rol se agrupan las herramientas que nos permiten la gestión del ecosistema.

- **YARN *(Yet Another Resource Negotiator)***

  Es el sistema operativo de datos para Hadoop 2 y es responsable de administrar el acceso a los recursos críticos de Hadoop.

Soporta múltiples motores de acceso a datos heterogéneos que son analizados con el procesamiento por lotes, consultas interactivas, búsquedas o técnicas de *Machine Learning*. Estos se pueden ejecutar simultáneamente y son gestionados de forma centralizada por YARN. La arquitectura de YARN desacopla el modelo de programación de la infraestructura de gestión de los recursos y delega muchas funciones programadas a los componentes de cada aplicación. En este nuevo contexto, *MapReduce* es una de las aplicaciones que se ejecuta en el nivel superior de YARN [34].

- **Ambari**

  Es una plataforma para la gestión y seguimiento de los clústeres de *Hadoop*. Incluye una colección de herramientas de operación y un conjunto de *RESTful API* que utilizan peticiones HTTP del tipo *GET, PUT, POST y DELETE* para operar los datos.

  El componente principal es la interfaz de usuario web, una interfaz utilizada para aprovisionar, administrar y supervisar los clústeres *Hadoop*. La interfaz de usuario web *Ambari* es la "cara" de la gestión de Hadoop. *Ambari* incluye el soporte para HDFS, *MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig* y *Sqoop* [35].

- **Zookeeper**

  Su función principal es la coordinación de las aplicaciones y servicios distribuidos. Un sistema distribuido debe ser capaz de llevar a cabo operaciones coordinadas teniendo en cuenta la escalabilidad, seguridad, consistencia, caídas de la red, las limitaciones de ancho de banda, y los problemas de sincronización. *ZooKeeper* está diseñado para dar soporte a estas necesidades [36].

- **Oozie**

  Es un motor de flujo de trabajo utilizado para ejecutar trabajos de Hadoop. Permite a los usuarios crear y programar transformaciones de datos complejas mediante la combinación de *MapReduce, Hive, Pig* y el uso de *Sqoop* en una sola unidad lógica de trabajo. Oozie también soporta la ejecución de *Java, Linux shell, distcp, SSH* y otras operaciones.

B. Aplicaciones para el componente proveedor de datos

- **HDFS (Hadoop Distributed File System)**

  Es un sistema que permite almacenar información desde un origen de datos hacia múltiples nodos [8]. Asimismo, opera bajo una arquitectura maestro/esclavo y está integrado por dos

componentes principales: *NameNode*, se encarga de manejar la información de los bloques de datos que hay en cada *DataNode*; es decir, almacena información acerca de cuantas veces un archivo ha sido replicado en el clúster y sobre cuantos bloques forman un archivo. Además, se encuentra el DataNode (esclavo), encargado del procesamiento de datos y del almacenamiento de los bloques actuales, ver Figura 3.1.

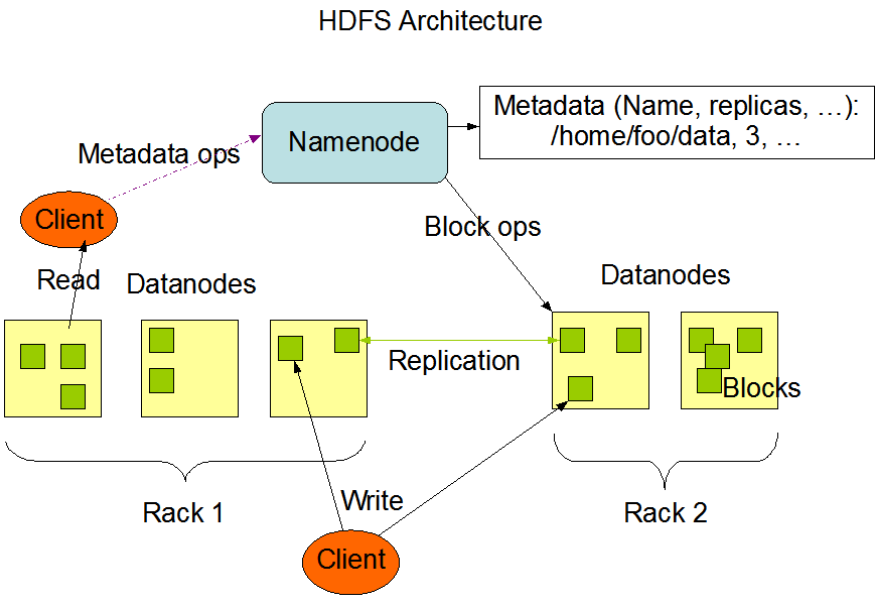


Figura 3.1. Arquitectura de HDFS

- **HBase**

  HBase es una base de datos no relacional o *NoSQL* orientada a columnas que se ejecuta en la parte superior de HDFS. A diferencia de las bases de datos relacionales, HBase no soporta SQL. HBase ofrece acceso aleatorio en tiempo real a los datos. Además, añade algunas capacidades transaccionales a Hadoop, lo que permite a los usuarios realizar inserciones, actualizaciones, búsquedas y eliminación [37].

- **Flume**

  Es un servicio que se encarga de recolectar, agregar y mover flujos de datos. Al ser un servicio distribuido se puede implementar en muchos sistemas. Es fiable, dado que su arquitectura y los componentes están diseñados para evitar la pérdida de datos. Es un servicio altamente disponible, porque utiliza la redundancia para limitar el tiempo de inactividad.

Flume está formado por entidades que son llamadas *sources*, *channels*, y *sinks*. Los *sources* pueden ser cualquier fuente de datos, dado que Flume cuenta con diversos adaptadores predefinidos. Un *sink* es el destino de una operación específica. Un *channel* es un almacén transitorio para eventos, donde los eventos son entregados al canal a través de fuentes que operan dentro del agente. Un evento puesto en un canal permanece en ese canal hasta que un *sink* lo elimina para un transporte posterior [38] [39].

- **Sqoop**

  Está formado por un conjunto de herramientas de importación y exportación. Sqoop es una herramienta diseñada para transferir datos entre Hadoop y los almacenes de datos estructurados como las bases de datos relacionales. Se puede utilizar Sqoop para importar datos estructurados externos en el sistema de archivos de Hadoop (HDFS) o a sistemas relacionados como Hive y HBase. Del mismo modo, se puede utilizar para extraer datos de Hadoop y exportarlos a los almacenes de datos estructurados externos tales como las bases de datos relacionales [40].

C. Aplicaciones para el componente proveedor de aplicaciones

En este rol se solapan otras aplicaciones del proveedor de datos ya que se considera en ambos a la actividad de recolección. Describiremos las aplicaciones que pertenecen a las actividades de preparación, análisis, visualización y acceso.

- **MapReduce**

  Es un modelo de programación utilizado por *Hadoop* para la distribución de tareas en los nodos del clúster y para el procesamiento distribuido de datos [29] [41]. La base de *MapReduce* está compuesta por dos funciones principales para el procesamiento de datos *Map* y *Reduce*. Un trabajo de *MapReduce* se encarga de dividir los datos de entrada en un conjunto de bloques independientes que luego son procesados en paralelo por la función *Map*. Una vez terminada la función *Map*, se ordena los resultados obtenidos por las funciones K y luego las envía para ser procesadas por la función *Reduce* (ver Figura 3.2). Mayormente, las salidas y entradas de datos son almacenadas por sistemas de archivos como por ejemplo HDFS. Además, la plataforma se encarga del monitoreo de las tareas programadas y la re-ejecución de las tareas fallidas.

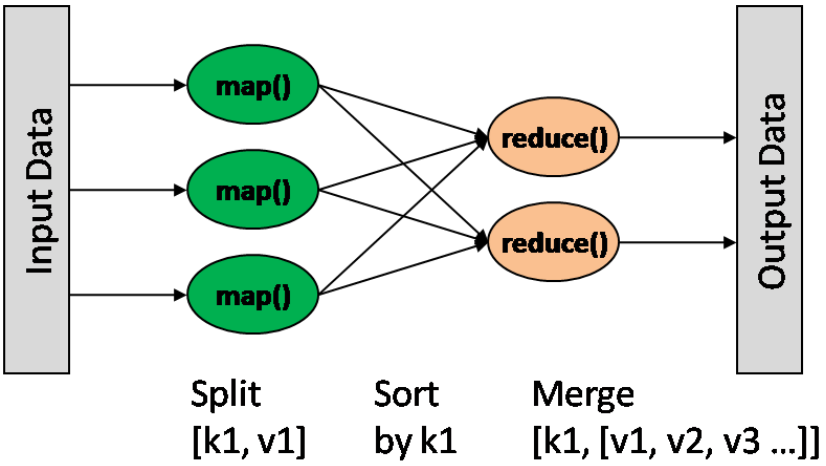


Figura 3.2. MapReduce

- **Mahout**

  Es una biblioteca de algoritmos de aprendizaje automático escalable, implementado en el nivel superior de Hadoop utilizando MapReduce. El aprendizaje automático *(Machine Learning)* es una disciplina de la inteligencia artificial que se enfoca en que las máquinas puedan "aprender" sin ser programadas de forma explícita y es utilizado, mayormente, para mejorar el rendimiento futuro basado en los resultados anteriores.

  Una vez que los grandes conjuntos de datos se almacenan en el sistema de archivos distribuidos de Hadoop (HDFS), Mahout brinda las herramientas para encontrar patrones significativos de forma automática. Mahout soporta cuatro casos de uso principales referente a la ciencia de los datos *(data science):*

  - El filtrado colaborativo: a través del análisis del comportamiento del usuario se obtienen recomendaciones de productos (por ejemplo, Amazon);
  - Agrupación *(clustering)*: se tienen elementos de una clase en particular y se organizan en grupos de tal manera que los elementos que pertenecen a un mismo grupo son similares entre sí;
  - Clasificación: se produce el aprendizaje a partir de categorizaciones existentes para luego asignar la mejor categoría a los elementos no clasificados; y
  - La minería en un conjunto de elementos frecuentes: es el análisis de elementos de un grupo para luego identificar los elementos que suelen estar juntos.

D. Servicios para el componente proveedor de infraestructura

Este componente está representado por la infraestructura interna o por servicios contratados en la nube como se ha detallado en la sección 2.3.1. La gestión de estos recursos se lleva a cabo con sus propias interfaces y procedimientos de acuerdo a la tecnología usada o al servicio contratado (AWS, Azure, etc). Por lo cual, en este caso no corresponde relacionar herramientas o aplicaciones del ecosistema Hadoop a este componente.

E. Aplicaciones para el componente consumidor de datos

- **Pig**

  Es una plataforma para la extracción, transformación y el análisis de grandes conjuntos de datos. Tiene dos componentes: el primero es el lenguaje que se llama PigLatin; y el segundo es un entorno de tiempo de ejecución, donde los programas PigLatin son ejecutados.

- **Hive**

  Es una infraestructura de almacenamiento de datos construida encima de Hadoop. Fue diseñado para permitir a los usuarios, con experiencia en el manejo de bases de datos relacionales, que realicen el análisis de los datos utilizando sentencias conocidas basadas en SQL. Hive incluye soporte para el análisis con el estándar SQL:2011. Esta plataforma permite tomar ventaja de los conocimientos y habilidades existentes de SQL en una implementación de Hadoop.

- **Solr**

  Es una plataforma de búsqueda e indexación de datos almacenados en HDFS. Solr proporciona búsquedas indexando los documentos a través de XML, JSON, CSV o HTTP para ayudar a encontrar los patrones de datos, relaciones y correlaciones.

- **QlikView**

  QlikView permite obtener analíticas personalizadas sobre los datos que residen en Hortonworks y para poder realizar la conexión con esta plataforma QlikView utiliza Apache Hive (vía conexión ODBC) como el acceso SQL a los datos en Hadoop. QlikView tiene dos modos de obtener datos: Datos cargados en memoria (Associative Data Store) que permite compresión de los datos y tiene un tiempo de respuesta muy rápido, y una solución híbrida (Direct Discovery) que permite analizar grandes cantidades de datos pero tiene un tiempo de respuesta mayor [42].

F. Aplicaciones para el componente de seguridad y privacidad

Este rol está presente en todos los niveles de la arquitectura desde la recolección de datos hasta la visualización de resultados. Las demás aplicaciones y servicios como HDFS, YARN, Hive también contribuyen a las características de seguridad de Hadoop.

- **Apache Knox**

  Es una puerta de enlace *(gateway)* perimetral que protege al clúster Hadoop. Proporciona un único punto de autenticación en un clúster Hadoop. Knox es capaz de proporcionar ayuda en el control, integración, supervisión y automatización de las necesidades administrativas y de análisis. Las políticas de seguridad que soporta Knox son:

  - Autenticación (LDAP y el proveedor de autenticación de *Active Directory*)
  - Federación / SSO (encabezado HTTP basado en la federación de identidades)
  - Autorización *(Service Level Autorization)*
  - Auditoría.

- **Apache Ranger**

  Es una plataforma de seguridad centralizada que ofrece controles de políticas para HDFS, Hive, HBase, Knox, Storm, Kafka y Solr. Permite el manejo de políticas para el acceso a ficheros, directorios, bases de datos, tablas y columnas. Estas políticas se pueden establecer para usuarios individuales o grupos.

3.2. Distribuciones Hadoop

En el ecosistema Hadoop existen diversas distribuciones que implementan una arquitectura común a diversas necesidades de Big Data. Estas integran un conjunto de herramientas de ingesta de datos, control de flujo, almacenamiento, análisis y procesamiento. A continuación, se detallan tres de las principales distribuciones que destaca el informe *"The Forrester Wave: Big Data Hadoop Distributions, Q1 2016"* [9]: Cloudera, Hortonworks y MapR. Estas distribuciones son las más usadas en los proyectos de Big Data.

3.2.1. Cloudera

Cloudera es una de las distribuciones de Hadoop más conocidas y usadas. Este agrega un conjunto de componentes propietarios al ecosistema Hadoop; estos componentes fueron diseñados para

optimizar la gestión de los clústeres y brindar mejores experiencias de búsquedas. Alguno de los componentes desarrollados por Cloudera son:

- Impala: es un motor basado en SQL, a tiempo real y paralelizado, realiza búsquedas de datos en el sistema de archivos HDFS. Es considerado como el motor de consultas más rápido de entre todos los proveedores de distribuciones Hadoop del mercado.
- Cloudera Manager: es la consola que permite gestionar y desplegar los componentes en el clúster Hadoop.
- Hue: es una consola que le permite al usuario interactuar con los datos y ejecutar *scripts* para los diferentes componentes de Hadoop contenidos en el clúster.

La Figura 3.3 ilustra la distribución Hadoop de Cloudera, sus componentes se clasifican de la siguiente manera:

- Componentes que son parte del núcleo de Hadoop: YARN y HDFS
- Componentes que son parte del ecosistema Hadoop: Zookeeper, Hbase, Hive, Pig, Flume, Oozie y Sqoop
- Componentes desarrollados por Cloudera: Cloudera Manager, Impala y Hue.

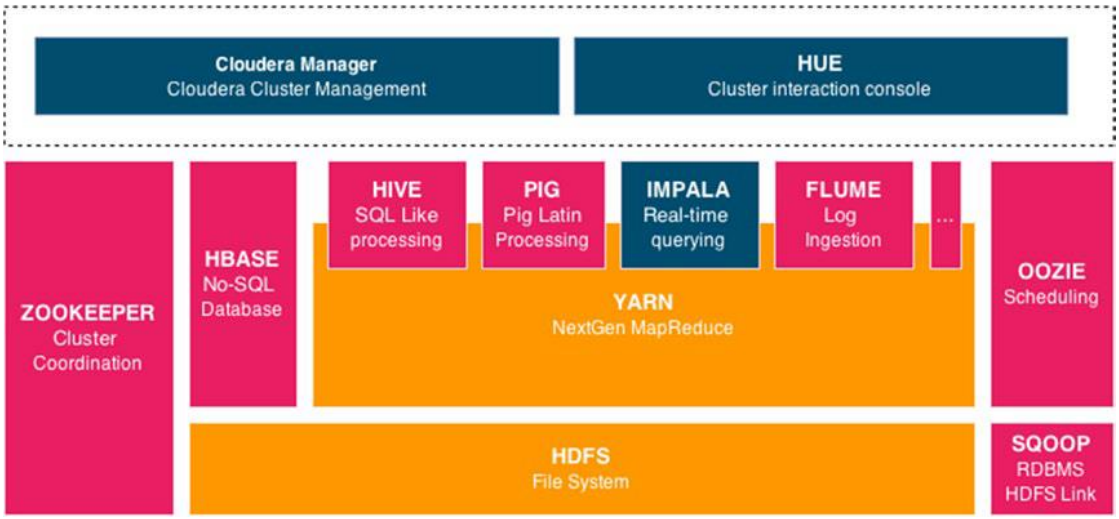


Figura 3.3. Distribución Hadoop Cloudera [43]

3.2.2. MapR

Es una distribución que tiene como pieza clave al sistema de archivos MapR. Implementa la API de HDFS, es de lectura/escritura completa y puede almacenar miles de millones de archivos (en comparación con la configuración compleja para HDFS que requiere espacios de nombres separados). Los usuarios de esta plataforma normalmente tienen o están planeando clústeres

Hadoop de misión crítica para lo cual pueden usar MapR-DB y MapR Streams (que implementan las APIs HBase y Kafka, respectivamente) [44]. En la Figura 3.4 se muestra las tecnologías que implementa MapR divididas en Motores de Ejecución y Gobierno de Datos y Operaciones.

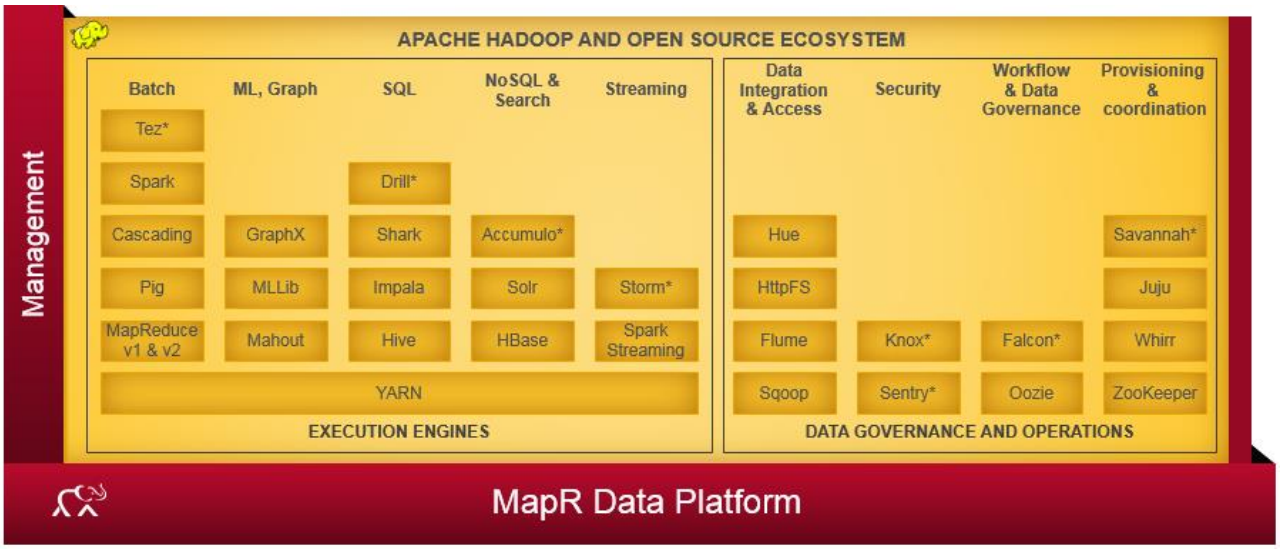


Figura 3.4. Distribución Hadoop MapR [44]

3.2.3. Hortonworks

Hortonworks ofrece una distribución cien por ciento de código abierto. Integra componentes estables del proyecto Hadoop antes que las últimas versiones lanzadas. Agrega *Ambari* como un componente de gestión en su arquitectura, el que es comparable con *Cloudera Manager*. Hortonworks posee las siguientes características:

- Implementación: en la infraestructura interna, en la nube o en una arquitectura híbrida; ya sea sobre plataformas *Linux* o *Windows.*
- Se recolectan y almacenan los datos en su forma original, sin importar su fuente o formato.
- Permite recoger datos de las bases de datos relacionales existentes, además de tipos de datos menos estructurados como el flujo de interacción o clics en sitios web *(web clickstream),* datos de máquinas y sensores, *social media*, datos desde móviles, geolocalización o los datos de registros de servidores.
- Tiene a YARN como su núcleo principal.

En la Figura 3.5 se puede apreciar los principales componentes que forman parte de esta distribución.

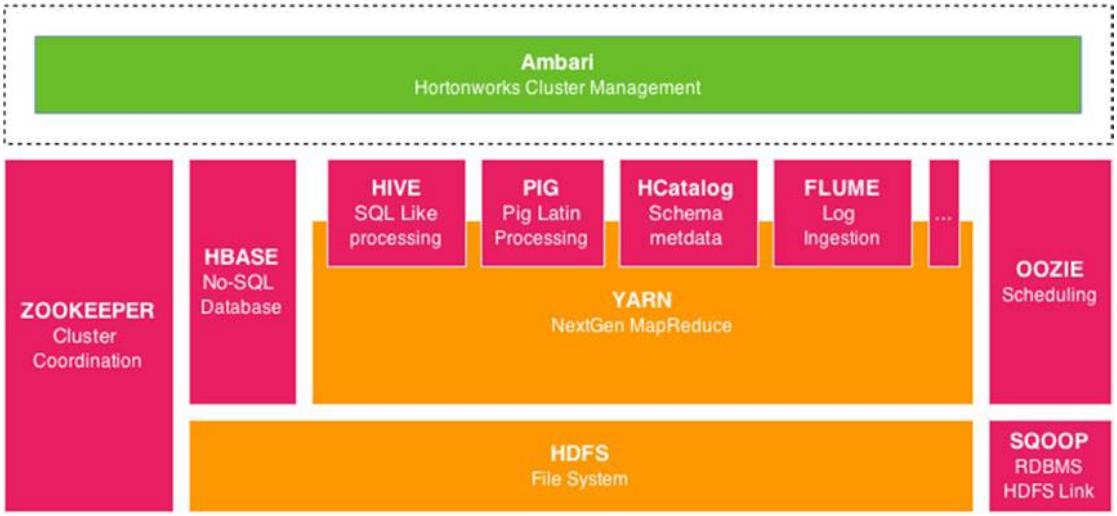


Figura 3.5. Distribución Hadoop Hortonworks [43]

3.3. Soluciones Cloud

Además, de las distribuciones Hadoop que pueden ser implementadas en la infraestructura interna de la organización, existen soluciones desplegadas de servicios en la nube *(cloud)* que, en muchos casos, pueden coincidir con implementaciones de dichas distribuciones. A continuación, detallaremos las dos plataformas más conocidas en este ámbito: *Amazon EMR (Elastic MapReduce)* y *Microsoft Azure*.

3.3.1. Amazon EMR *(Elastic Map Reduce)*

Esta plataforma permite analizar y procesar grandes cantidades de datos mediante la distribución del trabajo de cómputo, a través de un clúster de servidores virtuales. El clúster es gestionado bajo la plataforma Hadoop, donde un nodo es designado como maestro para controlar la distribución de tareas.

El servicio *Amazon EMR* ha realizado mejoras y personalizaciones en los componentes de Hadoop para trabajar de forma integrada con *Amazon Web Services (AWS)*. Los clústeres Hadoop que se ejecutan en *Amazon EMR* usan las siguientes instancias: *Amazon Elastic Compute Cloud* (Amazon EC2) que consiste en servidores virtuales Linux para los nodos maestro y esclavo, *Amazon Simple Storage Service (Amazon S3)* para el consumo y almacenamiento masivo de datos y *CloudWatch* para supervisar el rendimiento del clúster y generación de alertas (ver Figura 3.6).

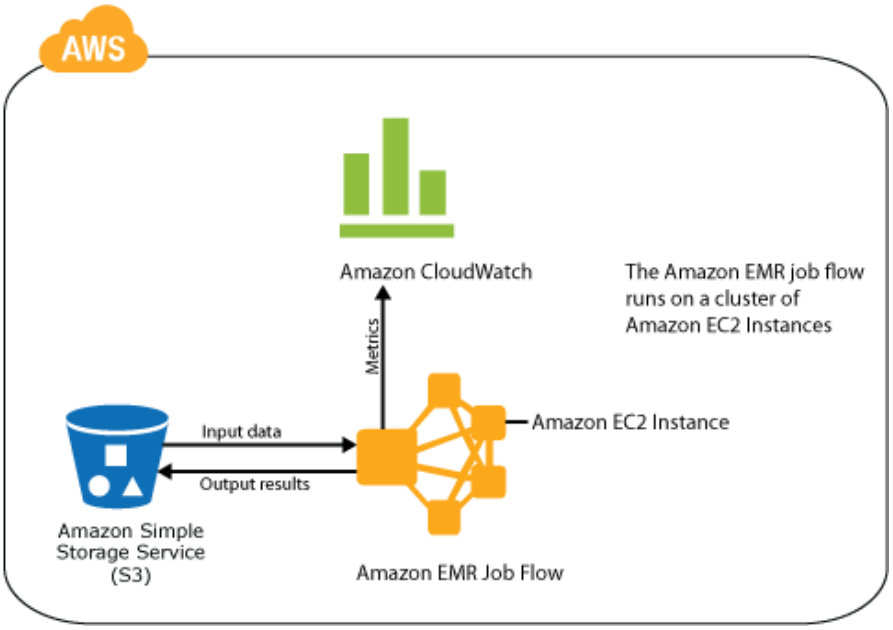


Figura 3.6. Interacción de Amazon EMR con otros servicios AWS [45]

3.3.2. Microsoft Azure

La plataforma *Azure* implementa una solución denominada *HDInsight* que utiliza la distribución *Hortonworks Data Platform (HDP)* basada en Hadoop [46]. Esta solución permite implementar y gestionar clústeres de Apache Hadoop en la nube con el fin de proporcionar un sistema óptimo para procesar, analizar y generar informes garantizando una alta confiabilidad y disponibilidad.

El Aprovisionamiento automático de clústeres de Hadoop: la creación de los clústeres de HDInsight es mucho más simple que la configuración manual de los clústeres de Hadoop. Para obtener información detallada, consulte sobre aprovisionamiento de clústeres de Hadoop en HDInsight.

A continuación, se describen las características más resaltantes de *HDInsight*:

- Constante actualización de los componentes Hadoop.
- Alta disponibilidad y confiabilidad de los clústeres.
- Almacenamiento de datos eficiente y económico con el almacenamiento de blobs [7] de Azure, opción compatible con Hadoop.
- Integración con otros servicios de Azure, como aplicaciones web y bases de datos SQL.
- Escalado de clústeres, que permite cambiar el número de nodos de un clúster de HDInsight en tiempo de ejecución.

- Compatibilidad con redes virtuales, lo que permite el aislamiento de los recursos en la nube o en escenarios híbridos para poder vincular los recursos de la nube con recursos locales.

3.4. Comparativa de las distribuciones y soluciones en la nube de Hadoop

Para poder determinar que distribución implementar dentro de un proyecto Big Data, es necesario evaluar las características del problema, así como, sus necesidades. Una vez realizado el análisis del proyecto a implementar y definida la arquitectura a utilizar, se deben evaluar las opciones de implementación disponibles que asemejen o implementen las herramientas necesarias para alcanzar el objetivo del proyecto.

Se debe considerar que al implementar un proyecto Big Data se ha de utilizar diversas fuentes de datos y, según sea el caso, crear diferentes almacenes de datos estructurados y no estructurados. Por ejemplo: Disney utiliza Cassandra, Hadoop y Mongo; Netflix utiliza Cassandra, Hbase, y SimpleDB; Twitter utiliza Cassandra, FlockDB, Hbase y MySQL; por citar algunos ejemplos [47].

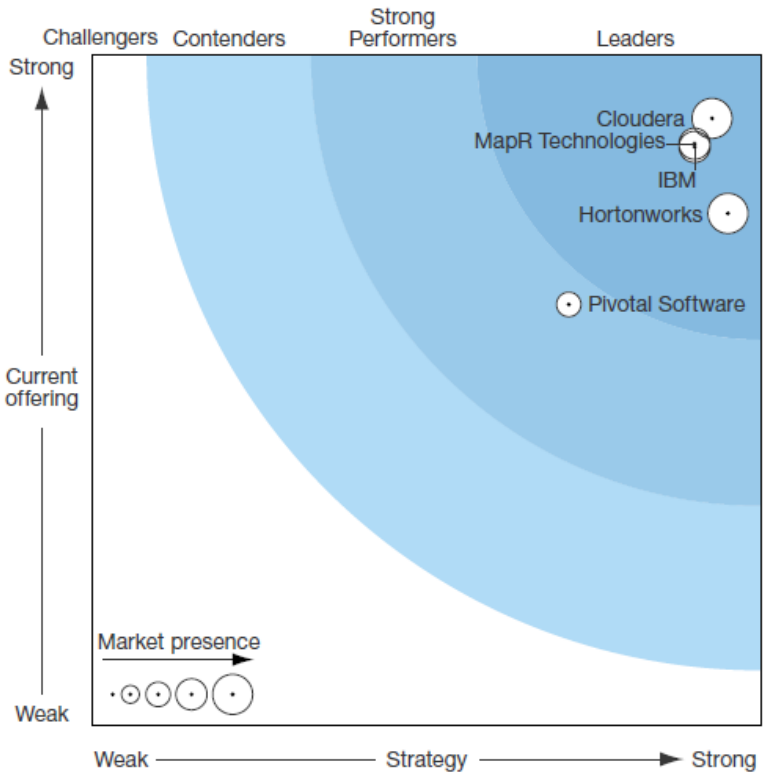


Figura 3.7. Top de distribuciones Hadoop. Fuente Forrester, 2016 [9]

Las tres distribuciones (Cloudera, Hortonworks y MapR) son equivalentes en la construcción de una arquitectura y tienen un alto grado de madurez lo que las convierten en plataformas líderes del mercado tal como se muestra en la Figura 3.7 del reporte elaborado por la empresa *Forrester Research* [9]. Sin embargo, algunas características como la de código abierto, en el caso de Hortonworks, elimina el riesgo de dependencia con un solo proveedor, fomenta la innovación de la plataforma y permite a los usuarios de Hadoop influir en su desarrollo y evolución. Todo esto hace de dicha distribución una opción, a tomar en cuenta, en la implementación de una solución Big Data.

Tabla 3.1. Comparativa de las soluciones Hadoop

Soluciones Hadoop	Ventajas	Desventajas
Cloudera	✓ *Cloudera Manager* e *Impala* consideradas como las mejores herramientas de gestión y motor de búsquedas, respectivamente. ✓ Soporta gran cantidad de herramientas del ecosistema Hadoop.	✓ Licencia comercial de la versión completa. ✓ En la versión CDH4 se tiene puntos únicos de fallo de los *namenode*. *HDFS HA* reduce los fallos, pero implica un mayor uso de recursos.
MapR	✓ Mejora el manejo de datos con un mayor rendimiento gracias a su sistema de archivos *MapRFS*. ✓ Elimina los puntos únicos de fallo. A diferencia de HDFS que si falla el *namenode* se pierde toda la información de los metadatos.	✓ Licencia comercial de la versión completa. ✓ Dependencia de los datos (data lock in).
Hortonworks	✓ 100% código abierto ✓ Soporte de Windows. ✓ Adaptación a las nuevas versiones de Hadoop. ✓ YARN como núcleo de gestión de datos.	✓ El soporte y mantenimiento puede llevar mucho esfuerzo y conocimiento.
Amazon EMR	✓ Integración con *Amazon Web Services (AWS)* ✓ Fácil crecimiento horizontal de la arquitectura según las necesidades. ✓ Permite quitar y agregar nodos según las necesidades de la carga de trabajo para el control de los costos.	✓ Dependencia con el proveedor ✓ Costos en el uso del servicio * ✓ Discutible protección de datos sensibles almacenados en el servicio.
HDInsights	✓ Basado en Hortonworks ✓ Almacenamiento mediante blobs de Azure, lo que elimina los puntos únicos de fallo. ✓ Fácil crecimiento horizontal de la arquitectura según las necesidades.	✓ Dependencia con el proveedor ✓ Costos en el uso del servicio * ✓ Discutible protección de datos sensibles almacenados en el servicio

En la Tabla 3.1, se hace referencia a las ventajas y desventajas de las soluciones basadas en Hadoop. En cuanto a la distribución de Hortonworks, por ser 100% código abierto nos permite

una adaptación más rápida a las nuevas versiones del ecosistema Hadoop y en el caso de proyectos de investigación, donde los recursos son limitados, el poder contar con una herramienta sin coste alguno es una ventaja a considerar. Así mismo, tiene el soporte nativo para Windows, que en muchos casos significa una rápida implementación de soluciones Big Data sin mayores cambios o adaptaciones de la infraestructura actual de las organizaciones. Es por estas características que se ha elegido a Hortonworks para implementar el prototipo del proyecto Bip4Cast.

Cloudera al ser una distribución con mayor madurez, ofrece herramientas personalizadas que sobresalen por su funcionalidad y eficiencia frente a las demás distribuciones. Aunque estas herramientas generalmente tengan una versión comercial y teniendo en cuenta que Cloudera es parcialmente de código libre, hace que se convierta en una buena opción cuando lo que se busca es productividad.

MapR ofrece un mejor manejo de los datos por su alta tolerancia a los fallos gracias a su sistema de archivos MapRFS. Este sistema elimina los puntos únicos de errores al no utilizar los *namenode*. Los usuarios de MapR normalmente buscan implementar clusters Hadoop de misión crítica con el uso de MapR-DB y MapR Streams.

Por último, las soluciones en la nube como *Amazon EMR* y *HDInsight de Microsoft Azure* que son los líderes del mercado según el reporte *"Big Data Hadoop Cloud Solutions, Q2 2016"* de Forrester [48], nos brindan la posibilidad de implementar una solución Big Data externalizada o híbrida por lo que se tiene que asumir costos por el uso de los nodos de un clúster en un tiempo determinado. Se debe considerar que al hacer uso de estos servicios se genera una alta dependencia con el proveedor. Del mismo modo, se debe tener muy en cuenta las políticas y condiciones en el tratamiento de los datos de dichas plataformas, las mismas que pueden infringir o atentar contra la normativa existente; como es el caso de la Ley de Protección de Datos de España [49], cuyas disposiciones deben ser consideradas en todas las soluciones de tratamiento de datos.

Capítulo 4. Arquitectura Big Data para el proyecto Bip4Cast

En el presente capítulo se plantea un prototipo de arquitectura diseñada para el funcionamiento óptimo del entorno Big Data para el proyecto Bip4Cast. La arquitectura tomará como base los componentes y herramientas definidas en los capítulos 2 y 3.

Los datos necesarios para el estudio provienen de diversas fuentes. Una es la que se genera con los actígrafos [4] [13] que posee cada paciente, estos dispositivos registran datos de la actividad física que son almacenados para luego ser volcados a un fichero, otra son los datos provenientes de una aplicación para móviles que recoge mediante un cuestionario información referente al estado de ánimo, consumo de bebidas, entre otras variables definidas por el estudio y por último los datos obtenidos desde un formulario web utilizado por el médico para registrar cada cita u observación realizada al paciente. Adicionalmente, se prevé contar con otras fuentes de datos como el registro de audio y vídeo para su comparación con los estados de eutimia, depresión e hipertimia.

Por lo que se ha clasificado al proyecto como Big Data principalmente por el volumen de datos que tiene que procesar, además de que se han de tener una variedad de fuentes de datos lo que será descrito detalladamente en las secciones siguientes.

4.1. Estado inicial de las fuentes de datos e infraestructura

En el planteamiento de la arquitectura Big Data para el proyecto Bip4Cast se realizó la evaluación del estado inicial, es decir, las características principales del problema, el estado de las fuentes de datos y la infraestructura informática con la que se cuenta.

4.1.1. Fuentes de datos

El estudio está diseñado inicialmente para 25 pacientes. Los pacientes han sido seleccionados entre los atendidos por las unidades de psiquiatría de la Clínica Nuestra Señora de la Paz, según los siguientes criterios de inclusión:

- Pacientes diagnosticados con trastorno bipolar.
- Resultados del tratamiento previo insuficientes (una o más descompensaciones anuales).
- Menos de 10 años desde el diagnóstico de la enfermedad.

- No se encuentran en situación de ciclación rápida [50].
- Edad entre 17 y 30 años.

Estos criterios seleccionan a pacientes graves con malos resultados en su tratamiento durante los primeros años de su enfermedad. En estos pacientes es más trascendente un tratamiento adecuado y una prevención precoz de las crisis.

Las fuentes de datos con las que se cuentan para el proyecto son las siguientes:

- Datos del Actígrafo: registro de la actividad física de cada paciente.
- Base de datos Oracle: con información general de cada paciente (edad, sexo, diagnóstico, etc.)
- Aplicación móvil: compuesta de un cuestionario que el paciente deberá responder diariamente. Estos datos se almacenan en una base de datos MongoDB.
- Aplicación web: que registra las observaciones del médico cuyos datos se almacenan en una base de datos MongoDB.

A. Estructura de datos del actígrafo

La recuperación de datos del actígrafo se realiza con la aplicación *GENEActive PC Software* que viene con el dispositivo [51]. Este software extrae un fichero con extensión .bin que contiene los datos capturados en hexadecimal (Ver Figura 4.1) por lo que es necesario utilizar la función de conversión para obtener un fichero .csv donde las primeras 100 filas pertenecen a la cabecera del archivo donde se tiene información como versión del *firmware*, frecuencia configurada, etc.

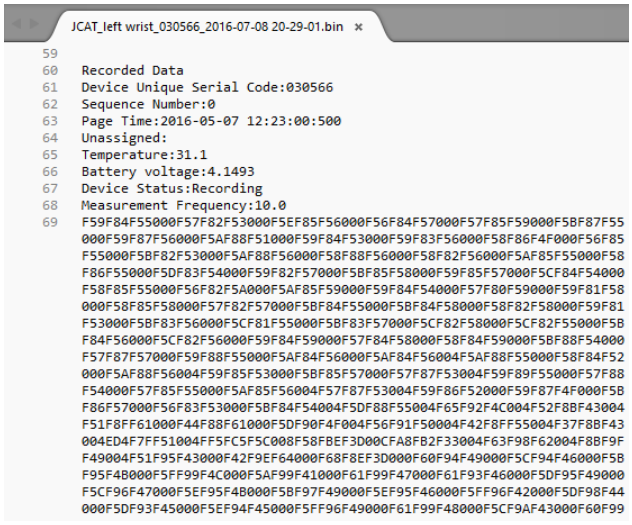


Figura 4.1. Datos capturados en el fichero .bin

En la Tabla 4.1 se muestra la estructura de los datos extraídos del actígrafo en fichero CSV a partir de la línea 101 y en la Figura 4.2 se puede apreciar los datos que contienen cada columna del fichero.

Tabla 4.1. Estructura de datos del fichero CSV

Columna	Dato
A	Marca de tiempo
B	Eje x (g) aceleración
C	Eje y (g) aceleración
D	Eje z (g) aceleración
E	Nivel de iluminación (lux)
F	Estado del botón (1/0)
G	Temperatura (°C)
H	Suma de magnitudes vectoriales
I	Desviación estándar del eje X
J	Desviación estándar del eje Y
K	Desviación estándar del eje Z
L	Pico de lúmenes

Tabla 4.2. Datos extraídos del actígrafo

timestamp	x	y	z	light	button	temperature
1457984001	-0.42392359	-0.35673706	-0.9593776	624.326531	0	25.3
1457984001	-0.22591917	-0.3724092	-0.99501128	507.265306	0	25.3
1457984001	-0.03179718	-0.29404851	-0.97917409	468.244898	0	25.3
1457984001	-0.23756649	-0.17650746	-0.88811023	526.77551	0	25.3
1457984001	-0.17544745	-0.21960585	-0.99105199	429.22449	0	25.3
1457984001	-0.24533137	-0.21960585	-0.93166251	487.755102	0	25.3
1457984001	-0.21038941	-0.18434353	-0.9554183	429.22449	0	25.3
1457984001	-0.25697869	-0.20785174	-0.93958111	409.714286	0	25.3

4.1.2. Infraestructura

La infraestructura se ha dividido en dos escenarios. El primero, es el de desarrollo y pruebas y el segundo, el de producción. En ambos escenarios se tiene un servidor similar, detallado en la Tabla 4.3. Además, se cuenta con el dispositivo principal en la recolección de datos: el actígrafo. Ambos se explican a continuación.

A. Características del Servidor

El servidor ha sido implementado con el sistema operativo Debian 7.1, dado que es una plataforma recomendada para la puesta en producción [52] de MongoDB. Esta base de datos

se utiliza para almacenar los registros que provienen de las aplicaciones web y móvil. Para poder montar la distribución Hortonworks se ha optado por la versión *Sandbox* que consiste en una máquina virtual que tiene integrada diversas aplicaciones. La máquina virtual donde se ha desplegado la *Sandbox* de Hortonworks es *VirtualBox.* Para poder subir los datos que recopilan los actígrafos se ha configurado un servicio SFTP [53] que agrega una capa de seguridad a la transmisión de archivos.

Tabla 4.3. Características del servidor de desarrollo

Servidor	Hardware	Software
Producción	**Procesador Intel Xeon E3-1246 Quad Core** **Memoria RAM 32GB DDR4** **Disco Duro 1TB 7200rpm** **Tarjeta de Red Gigabit Ethernet**	Sistema Operativo Debian 7.1 Base de Datos MongoDB Virtual Box Hortonworks Sandbox Servicio SFTP
Prueba	**Procesador Intel Core i7-2600** **Memoria RAM 16GB DDR3** **Disco Duro 1TB 7200rpm** **Tarjeta de Red Gigabit Ethernet**	Sistema Operativo Debian 7.1 Base de Datos MongoDB Virtual Box Hortonworks Sandbox Servicio SFTP

B. Características del Actígrafo

GeneActiv [51] es el Actígrafo elegido para el proyecto, después de haber evaluado diversas alternativas presentes en el mercado. Es de código abierto y ha sido validado para investigaciones científicas [54]. Tiene una duración de batería de 7 hasta 45 días de registro continuo, esto según se asigne la frecuencia, y cuenta en su construcción con material apto para no causar daños a la piel (ver Tabla 4.4), siendo esta una de las principales características para su elección.

Tabla 4.4. Características del actígrafo GENEActiv [18]

Parámetros Físicos	
Tamaño	43mm x 40mm x 13mm
Material	PC/aBS (grado de dispositivo médico)
Tipo de Batería	Polímero de litio recargbale
Protección del ambiente	
Humedad	Resistente al agua hasta 10m (iP67 – 1m 24hrs)
Materiales	Antipolvo (iP67)
Temperatura de operación	5 a 40 grados C
Impacto Mecánico	Resistente a caídas de 0.5m
Capacidades de Medición	
Memoria	0.5Gb
Frecuencias de Registro	10-100Hz

Periodos de Registro	45 días @10hz, 7 días @100hz
Reloj Interno	
Tipo	Reloj de Cuarzo de tiempo real
Frecuencia	32.768khz
Exactitud	+/- 20ppm (+/- 1.7s por día)
Mediciones de Aceleración	
Tipo de Sensor	MeMS
Rango	+/- 8g
Resolución	12 bit (3.9 mg)
Mediciones de Iluminación	
Tipo de Sensor	Fotodiodo de silicio
Rango	0 - 3000 Lux típica
Resolución	5 Lux típica
Exactitud	+/- 10% @ 1000 Lux de calibración
Registro de Eventos	
Tipo de Sensor	Interruptor de membrana mecánica
Mediciones de Temperatura	
Tipo de Sensor	Termosistor Lineal Activo
Rango	0 a 60 grados C
Resolución	0.25 grados C
Exactitud	+/- 1 grados C
Frecuencia de Medición	Cada 30s mínimo
Conexión USB	
Dispositivo	USB 2.0
Base de Carga	Formato de 4 unidades USB 2.0 *high speed*
Tiempo de Carga	90% @ 2 horas, 100% @ 3 horas
Tiempo de Descarga de Datos	Máximo 15 minutos por 4 unidades concurrentes

- **Configuración del actígrafo**

  La configuración para la captura de datos desde el Actígrafo se basa en diversas investigaciones tanto de validación de los sensores del GENEActive [54], como de otros artículos de investigación relacionados con el estudio del sueño [55]. En la Tabla 4.5 se muestra la relación entre la frecuencia y el periodo máximo de días de medición estos valores son prestablecidos por el software del actígrafo al momento de su configuración.

Tabla 4.5. Relación de frecuencia y periodo máximo de medición

Frecuencia de Medición (Hz)	Periodo máximo de Medición (Días)
10	45
20	30
25	28
30	21
40	18
50	15
60	12
66.7	11
75	10
85.7	**9**
100	7

Dado que el estudio requiere una mayor precisión de la actividad física del paciente, sobre todo para el análisis del sueño [16], se han considerado los siguientes valores de configuración en la toma de datos (ver Tabla 4.6):

Tabla 4.6. Valores de configuración del Actígrafo

Característica	Valor
Frecuencia de Medición	85.7 Hz
Período de Captura de datos	9 días
Localización del Acelerómetro	Muñeca Izquierda
Modo de Inicio del Acelerómetro	Inmediato
Lateralidad	Derecha/Izquierda (según sea el caso)

4.2. Definición de la arquitectura

En los capítulos anteriores se han detallado las diferentes capas de una arquitectura Big Data, los tipos de arquitectura y los casos de uso donde se aplican, además de las distribuciones Hadoop que son utilizadas para la implementación de múltiples arquitecturas. Diversos casos de uso de Big Data demuestran que no existe un modelo fijo, cada uno con distintas prioridades, por ejemplo, el procesamiento de datos por streaming o por lotes [14], [19], [23], [56].

Con el fin de asumir una referencia de arquitectura que esté sustentada en los estudios de varios dominios de Big Data y en las principales características que debe cumplir una arquitectura, se ha tomado como referencia a la arquitectura del grupo de trabajo del NIST [5] descrita en el capítulo 2.

4.2.1. Características principales del proyecto

Las características principales del proyecto se muestran en la Tabla 4.7 de acuerdo a la plantilla usada para el estudio de casos de uso por el NIST [24].

Tabla 4.7. Identificación de características del proyecto

Título	Predicción de Crisis en el trastorno bipolar "BIP4CAST"	
Área	Salud	
Actores/ *Stakeholders*	**Especialistas en salud mental** (Psicólogos, Psiquiatras): soporte al proyecto en cuanto a definición de protocolos de tratamiento y seguimiento del paciente, que brinden las características de la enfermedad para poder definir variables críticas. **Soporte de TI**: que brinde el aprovisionamiento y despliegue de los recursos necesarios para el proyecto. Tanto en procesamiento, almacenamiento y comunicaciones. **Especialistas en Hadoop**: conocimientos en la implementación de la plataforma Hadoop (distribuciones), así como del ecosistema necesario para el proyecto. **Analista de Datos**: mediante el uso de algoritmos, herramientas de análisis y otras técnicas, obtenga los patrones requeridos por el proyecto. Y que previamente realice la limpieza necesaria de los datos. **Especialista en seguridad**: que garantice la privacidad y seguridad de los datos en todo su ciclo de vida.	
Objetivos	Se pretende predecir mediante actigrafía y otras fuentes de datos la aparición de crisis en el trastorno bipolar con un sistema de alarma que notifica al paciente y a su médico de la proximidad de la crisis.	
Soluciones	**Almacenamiento**	HDSF, Oracle, MongoDB
	Redes	Red interna Gigabit Ethernet, Internet
	Software	Hadoop, Linux Debian, VirtualBox.
Características Big Data	**Fuentes de datos**	Registros de los actígrafos, aplicación web, aplicación móvil, BD local.
	Volumen (tamaño)	Se estima que el tamaño de almacenamiento durante 27 días y con 25 pacientes será de 58.6GB. Si se extiende a 2500 pacientes se llegaría a tener 6TB mensuales.
	Velocidad	Los actígrafos actualmente recolectan los datos en 9 días para luego subirlos a la plataforma para su procesamiento. Por lo que el procesamiento por lotes es aplicado en esta arquitectura. No obstante, esta debe tener soporte para Streaming en caso se implemente dispositivos que transmitan en tiempo real.
	Variedad	Se tienen datos en texto plano de los actígrafos, datos del historial clínico de en una base de datos Oracle, evaluaciones del médico en una aplicación web y una app móvil con un cuestionario diario ambos almacenados en MongoDB.
	Variabilidad	El cambio de modelo de actígrafos puede originar nuevos formatos de datos y la misma velocidad a la que se entregan.
Ciencia Big Data (colección, curación, análisis, acción)	**Veracidad**	Los datos capturados por los actígrafos pueden estar predispuestos a fallos sobre todo en la trasmisión SFTP, ante esto siempre existe una copia en el ordenador desde donde se hizo el envío. Además, estos datos una vez almacenados por SFTP son inmediatamente volcados en HDFS.
	Visualización	El volumen de entrada de datos, su exactitud e integridad deben ser controlados de forma rutinaria usando métodos de visualización. Ej. Amabri
	Calidad de Datos (sintaxis)	Los datos recolectados desde los actígrafos presentan un formato uniforme. Pero si el proyecto se extiende a otros dispositivos se tiene que plantear la normalización de dichos datos.
	Tipos de Datos	El proyecto incluye datos estructurados en texto plano, de bases de datos relacionales y NoSQL. Se tiene previsto la integración de audio y vídeo para reforzar el sistema de predicción comparando el estado eutímico del paciente con el estado en tiempo real.
	Analítica de Datos	Algoritmos de reconocimiento de patrones, análisis del sueño, clasificación, *clustering*, *Machine Learning* y *Boostrap* [57] para remuestreo.
Consideraciones en Movilidad	El proyecto cuenta con el desarrollo de una aplicación móvil que se integre a la arquitectura para mostrar las alertas de un evento de crisis bipolar tanto para el paciente como para el médico.	
Seguridad y Privacidad	La privacidad y confidencialidad de los individuos deben ser preservadas en el cumplimiento de los requisitos de las normativas existentes al existir datos de carácter personal y datos médicos	

4.2.2. Relación entre los requisitos de Big Data y la implementación Bip4Cast

En la Tabla 4.8 se muestra el conjunto de servicios y características que presenta el proyecto Bip4Cast en relación a los requisitos de Big Data detallados en la sección 2.1. Esto con el fin de corresponder la herramienta o aplicación de Hadoop con el cumplimiento de los requisitos de Big Data para de esta manera poder seleccionar adecuadamente dichas herramientas.

Tabla 4.8. Relación entre requisitos Big Data y Bip4Cast

Requisitos Big Data	**Bip4Cast**	
	Característica	**Herramienta / Servicio**
Requisitos de Fuentes de Datos		
DSR-1: Procesamiento de recolección.	Procesamiento por lotes	SFTP, Flume
DSR-2: Transmisión de datos.	Envío de ficheros	
DSR-3: Variedad de fuentes de datos.	Texto, audio y vídeo	HDFS
Requisitos del Proveedor de Transformación		
TPR-1: Técnicas de cálculo y procesamiento.	Limpieza de datos	R, MapReduce
TPR-2: Procesamiento analítico.	Procesamiento por lotes	MapReduce, YARN
TPR-3: Procesamiento de variedad de tipos de datos.	Texto, audio y vídeo	R, MapReduce, Solr
Requisitos del Proveedor de Capacidades		
CPR-1: Soporte a paquetes de software.		YARN
CPR-2: Soporte a plataformas de computación distribuidas.		Hadoop
CPR-3: Soporte para transmisión de datos.		Internet, LAN
CPR-4: Soporte para almacenamiento de datos distribuido.		HDFS
Requisitos del Consumidor de Datos		
DCR-1: Búsquedas de alta velocidad.		Hive
DCR-2: Visualización, representación y reportes.		R, QlikView
DCR-4: Interfaces de usuario.		Android, Web
Requisitos de Privacidad y Seguridad		
SPR-1: Soporte para datos sensibles.		SSH, Ranger, Knox, HDFS
SPR-2: Soporte para la autenticación y control de acceso.		SSH, Ranger, Knox
Requisitos de Gestión del Ciclo de Vida		
LMR-1: Calidad de datos.		Ambari HDFS MapReduce R Liberías, APIs YARN
LMR-2: Actualizaciones datos y perfiles de usuario.		
LMR-3: Ciclo de vida de los datos.		
LMR-4: Validación de datos.		
LMR-5: Prevención de pérdida de datos.		
LMR-7: Archivos en múltiples sitios.		
LMR-8: Identificador de datos y seguimiento.		
LMR-9: Estandarización, agregación, normalización de datos		
Otros Requisitos (Plataformas móviles)		
OR-1: Para acceder a los resultados	En desarrollo	
OR-2: Para monitorización del procesamiento analítico		
OR-3: Búsquedas		
OR-4: Adquisición de datos		
OR-5: Seguridad a través de los dispositivos móviles.		

4.2.3. Principios a cumplir por la arquitectura Bip4Cast

En el diseño de la arquitectura Big Data para el proyecto de predicción de crisis bipolar se han establecido los siguientes principios a cumplir para su implementación [12]:

Flexibilidad y escalabilidad: la arquitectura debe ser capaz de dar cobertura a la incorporación de nuevas fuentes de datos y permitir el crecimiento horizontal, dentro de la infraestructura interna o en caso se decida migrar a un servicio externo en la nube.

Inmutabilidad de los datos: los datos capturados del paciente, sobre todo los que provienen de los actígrafos, se almacenarán en un repositorio donde no serán modificados, llevando una copia de estos datos al sistema de procesamiento lo que garantiza que siempre se pueda regenerar los datos transformados en caso de error.

Tiempo de predicción: mientras menor sea el tiempo de obtención, procesamiento y análisis de datos, obtendremos mejores resultados en la predicción de las crisis por paciente.

Gestión de los datos: lograr un modelo que asegure la integridad y validez de los datos; y brinde niveles de seguridad adecuados para el tratamiento de la información sensible de los pacientes por medio de la autenticación, autorización, cifrado y anonimización.

4.3. Modelo de la arquitectura Big Data para el proyecto Bip4Cast

Una vez definidos los componentes principales de una arquitectura de acuerdo al capítulo 2, las herramientas disponibles para su implementación descritas en el capítulo 3 y especificación de las características principales del proyecto (desarrolladas en el presente capítulo), se procede a diseñar una arquitectura Big Data considerando los temas planteados.

En la Figura 4.2 se define la arquitectura que da soporte al proyecto Bip4Cast, donde se especifican los elementos que participan en la generación, captura, almacenamiento y análisis de datos en un entorno Hadoop. La implementación de la arquitectura se ha desplegado considerando las características del proyecto definidas en la Tabla 4.7 y 4.8, bajo la distribución Hadoop Hortonworks que ha sido descrita en la sección 3.2.3. Esta distribución nos brinda la integración de diversas plataformas de recolección de datos, almacenamiento HDFS y procesamiento con el paradigma MapReduce, todo coordinado bajo YARN [58], cubriendo gran parte de la arquitectura propuesta.

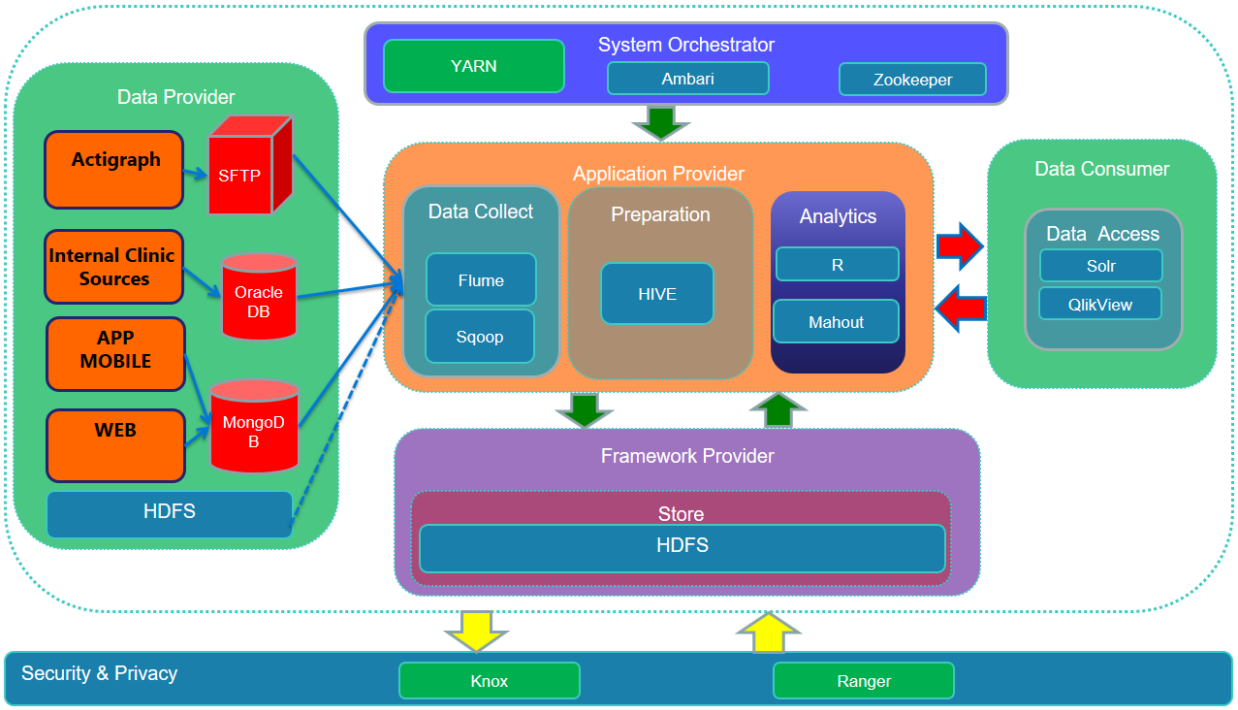


Figura 4.2. Arquitectura Big Data para el proyecto Bip4Cast

Como se aprecia en la Figura 4.2 la arquitectura queda agrupada en seis bloques que pertenecen a los componentes de la arquitectura definida por el NIST donde:

- Proveedor de Datos: contiene a las diversas fuentes de datos incluyendo a HDFS debido a que este también cumple la función de proveedor de datos para los demás componentes, en cuanto a los datos de los actígrafos se plantea la creación de un servicio SFTP para poder volcarlos al servidor.
- Proveedor de Aplicaciones: contiene los servicios de recolección de datos donde Flume permite enviar los ficheros entregados por el servicio SFTP hacia HDFS, la preparación donde Hive nos permite preparar los datos para ejecutar consultas y el análisis, la integración con R para aprovechar las ventajas de este lenguaje en el proyecto ya que cuenta con librerías para procesar los ficheros de los actígrafos, además de contar con Mahout que permite ejecutar algoritmos de *Machine Learning*.
- Consumidor de Datos: donde se tiene a Solr para poder realizar búsquedas dentro de HDFS y a QlikView como una herramienta de visualización para poder mostrar los resultados obtenidos del procesamiento de los datos en Hadoop.

- Sistema de Orquestación: se tiene a YARN como el sistema de gestión de aplicaciones en Hadoop, Ambari como la interfaz de gestión de la arquitectura y Zookeeper que ofrece un servicio de coordinación de procesos distribuido.
- Proveedor de Infraestructura: donde se tiene los recursos necesarios para poder implementar la arquitectura Big Data, se resalta el almacenamiento dado que será HDFS donde se vuelquen los datos recolectados.
- Seguridad y Privacidad: se tiene a Ranger que nos permite implementar políticas de seguridad en cada uno de los componentes de Hadoop y a Knox que brinda seguridad perimetral a Hadoop.

Los detalles de implementación del prototipo de arquitectura planteado se muestran detalladamente en el siguiente capítulo.

Capítulo 5. Implementación de la arquitectura Bip4Cast

En el presente capítulo se va a describir el proceso de implementación de la arquitectura en un entorno Hadoop utilizando la distribución Hortonworks. La arquitectura cuenta además con un servicio SFTP que fue implementado para poder realizar la carga de los datos del actígrafo en un servidor que implementará la solución Big Data. Estas implementaciones y otras características serán detalladas en las secciones siguientes.

5.1. Implementación del servicio SFTP para la recolección de datos

En este trabajo, nos centramos en la mayor fuente de datos que proviene de los actígrafos que llevan los pacientes las 24 horas del día y permite la recolección de datos por más de una semana.

Con el fin de asegurar la integridad de los datos generados por los actígrafos se ha determinado implementar el servicio SFTP. En las secciones siguientes, se va a describir la instalación y configuración del servicio SFTP en el servidor Linux Debian. *SFTP (Secure File Transfer Protocol)* es una extensión de la *SSH (Protocolo Secure Shell)* que se utiliza comúnmente para el acceso remoto con una capa de seguridad.

Para la recolección de datos del actígrafo se ha desarrollado y diseñado un modelo que consiste en una aplicación cliente que permite subir los datos a un servidor a mediante un servicio SFTP (ver Figura 5.1).

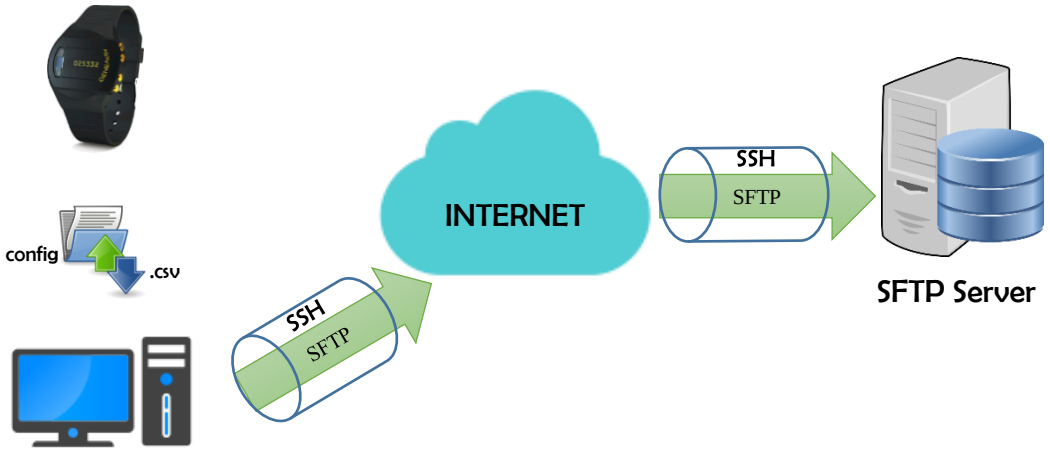


Figura 5.1. Modelo de Recolección de Datos mediante SFTP

5.1.1. Configuración del servicio SFTP

A continuación, se describen los pasos realizados en el servidor *Linux Debian* para implementar el servicio SFTP, por el cual los usuarios enviaran los archivos con los datos extraídos de los actígrafos.

Paso 1: Instalación del servidor OpenSHH

sudo apt-get install openssh-server

Paso 2: Creación de grupo y usuario

Creamos un grupo de usuarios para el acceso SFTP, que denominamos "sftppcb". Para mayor seguridad, no permitir que las cuentas con acceso SFTP puedan ingresar al servidor utilizando Secure Shell (SSH).

sudo groupadd sftppcb

Luego agregamos el usuario de uso exclusivo para el acceso SFTP.

sudo useradd usersftp -d / -g 1001 -M -N -u 1001
sudo passwd usersftp

Paso 3: Configuración del servicio SFTP

Realizamos una copia de seguridad del archivo de configuración del demonio SSH.

sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak

A continuación, realizaremos algunos cambios en el archivo de configuración de SSH.

sudo nano /etc/ssh/sshd_config

La línea *Subsystem sftp /usr/lib/openssh/sftp-server* debe ser reemplazada por ***Subsystem*** *sftp internal-sftp*

Después de *"UsePAM Yes"* agregamos las siguientes líneas para establecer los permisos y configuraciones del grupo "sftppcb". La opción *ChrootDirectory* hará que los usuarios del grupo "sftppcb" solo puedan tener acceso a un directorio en específico. De lo contrario, el grupo "sftppcb" podría tener acceso a la raíz del servidor y es algo que se debe evitar. Usaremos como directorio *ChrootDirectory* a */var/sftpfiles*.

Match group sftppcb
ChrootDirectory /var/sftpfiles
X11Forwarding no
AllowTcpForwarding no
ForceCommand internal-sftp

Ahora se crea el directorio asignado en ChrootDirectory: sudo mkdir /var/sftpfiles/

El directorio y sus directorios padre que son asignados en *ChrootDirectory* deben ser propiedad de *root* y asignados al grupo *root*. De lo contrario, los usuarios SFTP no podrán subir o modificar archivos y directorios.

Paso 4: Asignación de permisos

Por último, dentro del directorio "*sftpfiles"* creamos otro directorio que será donde se cargarán los datos de los actígrafos. Además, se asignan sólo los permisos necesarios para el usuario que subirá los datos.

mkdir userdata
chown root:sftppcb userdata
chmod 755 userdata

Los permisos especificados para el directorio permiten la escritura, pero no la lectura, lo que asegura que los datos no puedan ser modificados para garantizar la inmutabilidad de los mismos.

5.1.2. Implementación de la aplicación cliente SFTP

Con el fin de recolectar los ficheros extraídos con la aplicación del actígrafo en un servidor central se ha desarrollado una aplicación cliente SFTP (ver Apéndice: Aplicación cliente SFTP con Python y PyQt). En la Figura 5.2 de puede apreciar la interfaz de la aplicación cliente SFTP.

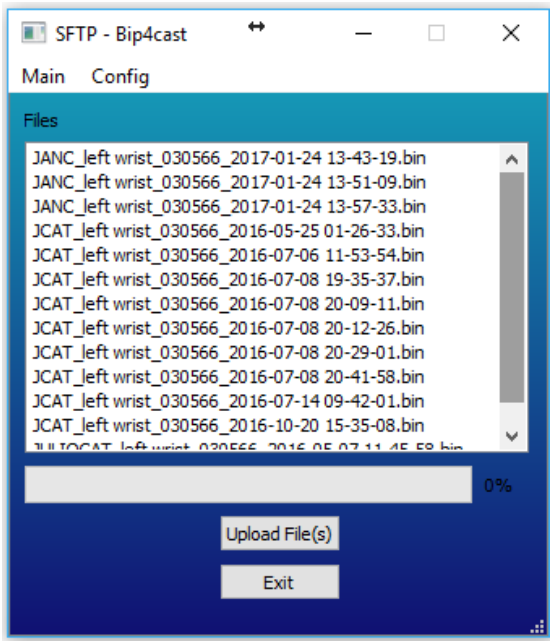


Figura 5.2. Interfaz gráfica de la aplicación cliente SFTP

La aplicación verifica los ficheros que se encuentran dentro del directorio por defecto donde se guardan después de ser extraídos del actígrafo, para poder enviar los ficheros se solicita la contraseña del usuario SFTP, una vez enviados los ficheros la aplicación modifica los nombres agregando el sufijo 'UPLOADED' (ver Figura 5.3) y los filtra del listado.

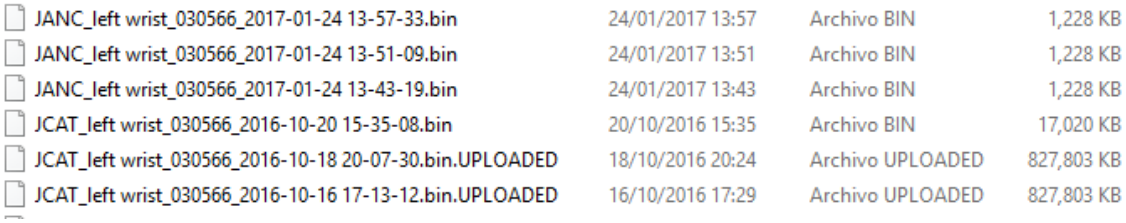


Figura 5.3. Ficheros extraídos del actígrafo

La aplicación ha sido desarrollada con el lenguaje de programación Python, la interfaz gráfica se ha logrado con QtDesigner [59] y la librería PyQt [60], la conexión al servicio SFTP con la librería pysftp [61] y la compilación para la distribución del ejecutable (Windows) se ha realizado con el paquete cx_Freeze [62] que nos permite generar distribuciones para Linux, Mac OS X y Windows.

Durante el desarrollo del proyecto se consiguió agregar la funcionalidad de enviar los archivos por SFTP a la aplicación del actígrafo *(GENEActiv PC Software)* para lo cual se ha utilizado la librería *SharpSSH* [63]. En la Figura 5.4 se muestra parte del código que permite el envío del archivo extraído al servidor y en la Figura 5.5 se muestra la interfaz donde se tiene el botón *"Send Data"* que ejecuta la funcionalidad de envío de los archivos. La codificación completa se encuentra en el Apéndice: Código para el envío de archivos por medio SFTP.

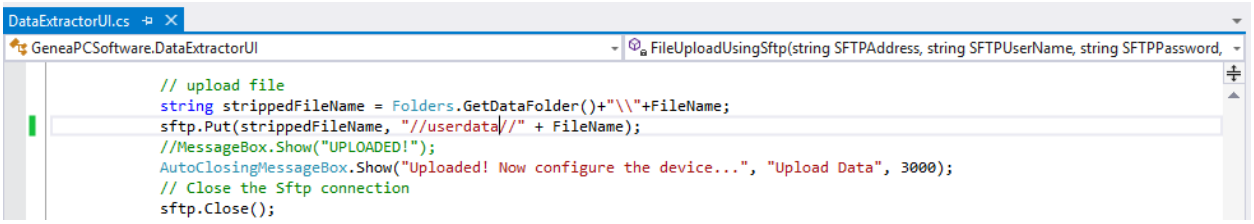


Figura 5.4. Porción de código para el envío de archivos mediante el servicio SFTP

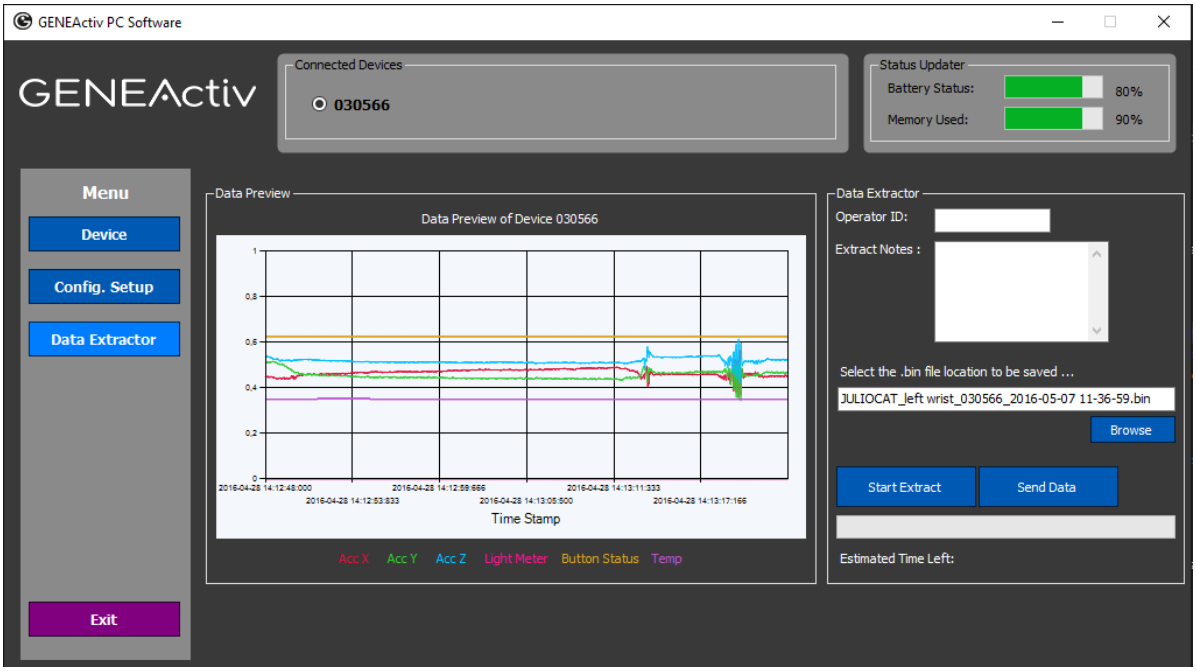


Figura 5.5. Interfaz de la aplicación del Actígrafo

5.2. Implementación del entorno Hadoop con Hortonworks

Como se ha indicado anteriormente, la arquitectura propuesta para el proyecto Bip4Cast ha sido implementada con la distribución Hortonworks descrita en la sección 3.2.3. La Sandbox Hortonworks nos brinda la posibilidad de tener instalado en una máquina de desarrollo una arquitectura Big Data con la que poder hacer pruebas sin necesidad de tener que instalar, configurar y mantener una arquitectura con los costes que ello supone.

De la arquitectura propuesta en la sección 4.3, los componentes de Hortonworks que serán utilizados y configurados son:

- recopilación de datos: *Flume* y *Sqoop*
- almacenamiento: *HDFS*
- análisis: *R* y *Mahout*
- visualización: *QlikView*
- consultas: *Hive*
- seguridad: *Ranger*
- coordinación y gestión: *YARN*, *Ambari*

En las subsecciones siguientes se describe más detalladamente cada uno de los pasos.

5.2.1. Implementación de la Sandbox de Hortonworks

Debido a que se necesita probar la arquitectura planteada para el proyecto Bip4Cast, en este caso se trata de un entorno en el que lo primordial es la capacidad de experimentación e investigación. Se ha diseñado una solución personalizable por lo que se busca que la plataforma sea lo más abierta posible.

Como primer paso, se ha de descargar la versión *Sandbox* de la distribución Hortonworks, en nuestro caso se ha utilizado la versión 2.4 disponible para *VirtualBox*. Después de implementar la SandBox con las consideraciones y los requisitos necesarios (ver Apéndice: Instalación de la SandBox Hortonworks) se dispone de interfaces web, como se aprecia en la Figura 5.6, que brindan acceso a los diversos servicios del entorno Hortonworks.

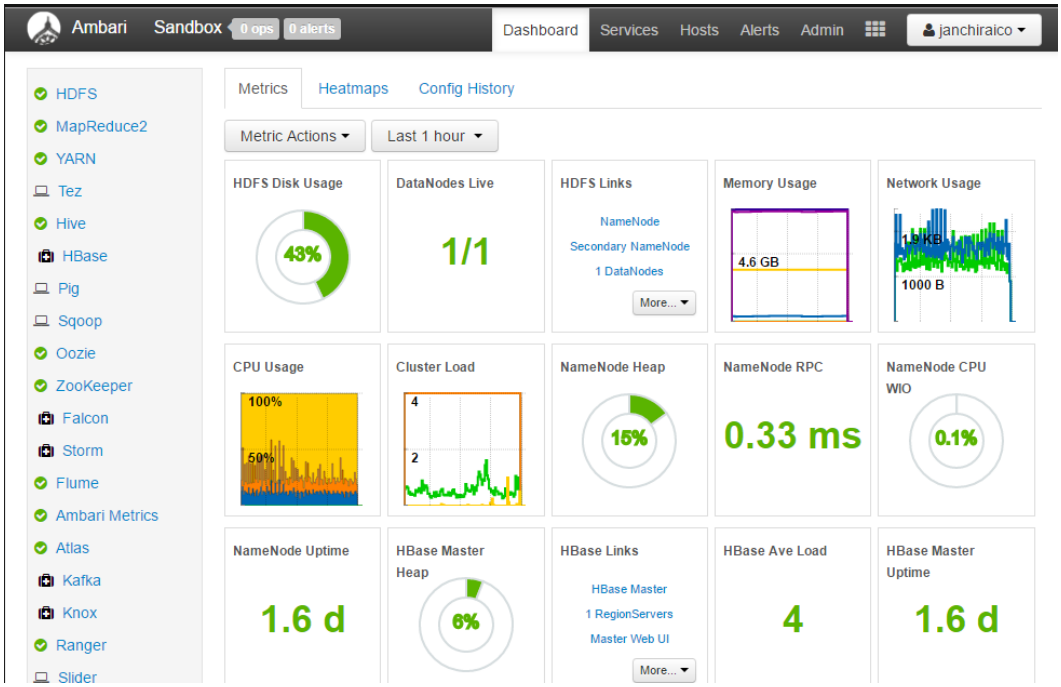


Figura 5.6. Interfaz web del servicio Ambari

Las siguientes son configuraciones importantes a tomar en cuenta en la implementación de la arquitectura Big Data y se pueden consultar en la documentación de Hortonworks [64]. En un clúster Hadoop, es vital equilibrar el uso de la RAM, la CPU y el disco para que el procesamiento no se vea limitado y evitar en lo posible cuellos de botella. Como recomendación general, se ha encontrado que permitir 1-2 contenedores por disco y por núcleo proporciona el mejor equilibrio para la utilización del clúster. Parte de la memoria RAM debe reservarse para el uso del sistema operativo.

Para calcular el tamaño de memoria que se asignará a YARN, MapReduce y JVM Heap [65] usaremos las especificaciones basadas en la determinación de los ajustes de configuración de memoria en un clúster con Hortonworks [66].

En nuestra máquina virtual contamos con 8 cores, 8GB de RAM y 1 disco. Para 8GB de RAM se recomienda reservar 2GB para el sistema operativo y 1GB para HBase dejándonos 5GB que serán utilizados por YARN. El número de contenedores está dado por la fórmula:

$$\#\text{Cnt} = [\min(2 * \text{cores}, 1.8 * \text{discos}, \text{TRAM} / \text{MIN\_CNT\_SIZE})]$$

Donde:

- #Cnt = número de contenedores
- cores = número de cores del nodo
- discos = número de discos del nodo
- TRAM = total de memoria RAM disponible
- MIN_CNT_SIZE = tamaño mínimo del contenedor (en RAM)

Aplicando la fórmula obtenemos:

$$\#\text{Cnt} = [\min\left(2 * 8, 1.8 * 1, \frac{5120}{512}\right)]$$

$$\#\text{Cnt} = [\min(16, 1.8, 10)]$$

$$\#\text{Cnt} = 2$$

Ajustamos #Cnt de 2 a 3 para no desaprovechar la utilización de los mismos de modo que se distribuyan en más contenedores.

Ahora como cálculo final obtenemos el tamaño de RAM por contenedor mediante la fórmula:

$$\text{RAMcnt} = \max(\text{MIN\_CNT\_SIZE}, (\text{TRAM}) / \#\text{Cnt}))$$

$$\text{RAMcnt} = \max(512, 5120/3)$$

$$\text{RAMcnt} = \max(512, 1706.66)$$

Ajustamos 1706.66 a un múltiplo cercano de 512 que es valor mínimo de un contenedor

$\frac{1706.66}{512} = 3$, entonces 512*3=1536

$$\text{RAMcnt} = 1536$$

En la Tabla 5.1 se definen los valores a ser configurados tomando en cuenta los cálculos realizados para YARN y MapReduce.

Tabla 5.1. Cálculos de valores de configuración para YARN y MapReduce

Fichero	Configuración	Cálculo del valor	Valor
yarn-site.xml	yarn.nodemanager.resource.memory-mb	= contenedores *RAM-por-contenedor	4608
yarn-site.xml	yarn.scheduler.minimum-allocation-mb	= RAM-por-contenedor	1536
yarn-site.xml	yarn.scheduler.maximum-allocation-mb	= contenedores *RAM-por-contenedor	4608
mapred-site.xml	mapreduce.map.memory.mb	= RAM-por-contenedor	1536
mapred-site.xml	mapreduce.reduce.memory.mb	= 2 *RAM-por-contenedor	3072
mapred-site.xml	mapreduce.map.java.opts	= 0.8 *RAM-por-contenedor	1228
mapred-site.xml	mapreduce.reduce.java.opts	= 0.8 *2 *RAM-por-contenedor	2457
mapred-site.xml	yarn.app.mapreduce.am.resource.mb	= 2 *RAM-por-contenedor	3072
mapred-site.xml	yarn.app.mapreduce.am.command-opts	= 0.8 *2 *RAM-por-contenedor	2457

A. Configuración de YARN

En nuestra máquina virtual contamos con 8 cores y 8GB de RAM, usaremos 5GB para YARN. La siguiente propiedad establece la memoria máxima que YARN puede utilizar en el nodo, se registra en el fichero yarn-site.xml:

<name>yarn.nodemanager.resource.memory-mb</name>
<value>4608</value> // 3 contenedores x 1536 de tamaño de RAM

El siguiente paso es indicar a YARN cómo dividir el total de recursos disponibles en contenedores. Para ello, se especifica la unidad mínima de RAM que se va a asignar a un contenedor. Si se quiere tener un máximo de 3 contenedores dividimos el total de memoria del nodo entre el número de contenedores lo que nos da 1.5 GB mínimo por contenedor.

<name>yarn.scheduler.minimum-allocation-mb</name>
<value>1536</value>

B. Configuración de MAPREDUCE 2

MapReduce 2 utiliza los contenedores YARN para programar y ejecutar las tareas Map y Reduce. Al configurar la utilización de recursos en YARN, hay tres aspectos a considerar:

- El límite de RAM físico para cada tarea de Map y Reduce
- El límite de tamaño del *JVM heap* para cada tarea
- La cantidad de memoria virtual de cada tarea

Debido a que cada Map y Reduce se ejecutarán en un contenedor independiente, estos valores máximos de memoria deberían ser mayor o igual a la asignación mínima (1.5GB) de los contenedores de YARN. Esto se registra en el fichero mapred-site.xml:

<name>mapreduce.map.memory.mb</name>
<value>1536</value>
<name>mapreduce.reduce.memory.mb</name>
<value>3072</value>

Cada contenedor realizará la ejecución de JVM para las tareas MapReduce. El tamaño de la *JVM heap* debe establecerse a un valor inferior de la memoria Map y Reduce definida, y así asegurar que se encuentre dentro de los límites de la memoria del contenedor YARN. Esta configuración se realiza en el fichero mapred-site.xml:

<name>mapreduce.map.java.opts</name>

<value>-Xmx1228m</value>

<name>mapreduce.reduce.java.opts</name>

<value>-Xmx2547m</value>

5.2.2. Implementación de la ingesta de datos con Flume

Después de subir los archivos con los datos del actígrafo al servidor SFTP, se deben mover al sistema de archivos distribuidos para su almacenamiento y procesamiento. La distribución Hadoop Hortonworks integra esta herramienta de ingesta de datos que permite mover grandes cantidades de datos por medio de una arquitectura basada en la transmisión de flujos de datos. Además de ser tolerante a fallos y de contar con mecanismos de recuperación [39], [67]. En la Figura 5.7 se puede apreciar los componentes del modelo implementado.

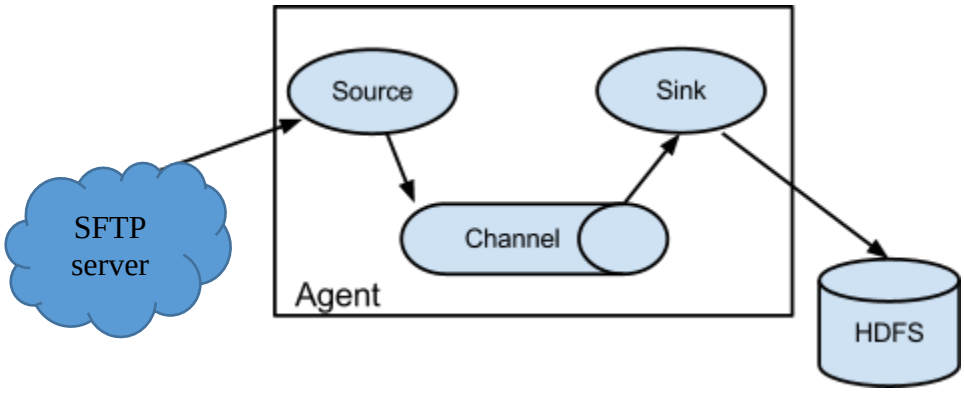


Figura 5.7. Arquitectura Flume

Se ha configurado Flume para que cuando un archivo es subido al directorio del servidor SFTP se copie inmediatamente al entorno Hadoop, es decir, se suba el archivo a HDFS. Se tiene configurado un agente Flume para que cuando se detecte que hay un nuevo archivo en el directorio del servidor SFTP éste se copie al directorio correspondiente en Hadoop. Al archivo original, una vez copiado, se le agrega el sufijo "COMPLETED" que es el valor por defecto.

A continuación, se muestra parte del archivo de configuración de Flume (ver Apéndice: Archivo de configuración de Flume) donde se ha utilizado el tipo de fuente *Spooling Directory* se

especifica el directorio de donde se recogerán los archivos (spoolDir), y a que directorio HDFS se volcarán (hdfs.path), estas son las configuraciones a tener presente al momento implementar un servicio Flume.

```
# Define a source for agent1
agent1.sources.source1_1.type = spooldir
# on linux FS
#Spooldir /var/sftpfiles/userdata
agent1.sources.source1_1.spoolDir = /var/sftpfiles/userdata
agent1.sources.source1_1.fileSuffix = .COMPLETED

#Sink is /userdata under hdfs
agent1.sinks.hdfs-sink1_1.hdfs.path = hdfs://localhost.localdomain:8020/userdata
```

Una característica que se no debe pasar por alto es la configuración del uso de memoria de Flume (dentro del fichero flume-env.sh), ya que se van mover archivos de tamaños mayores a la configuración por defecto.

```
export JAVA_OPTS="-Xms512m -Xmx1024m -Dcom.sun.management.jmxremote"
```

5.2.3. Implementación de importación bases de datos relacionales con Sqoop

Sqoop será usado para obtener datos de una base de datos relacional implementada sobre Oracle. Se volcarán los registros de las tablas que se consideren necesarias para ser procesados en el entorno Hadoop. Estas tablas contienen datos de los pacientes diagnosticados con el trastorno de bipolaridad (ver Apéndice: Modelo relacional de la base de datos PCB). La configuración para realizar esta operación con Sqoop requiere los siguientes pasos:

- Copiar el archivo JAR del driver Oracle Database JDBC a la *classpath* de *Sqoop*.
  sudo mv $ORACLE_HOME/jdbc/lib/ojdbc14_g.jar /usr/hdp/current/sqoop-client/lib

- Ejecutar la siguiente orden para volcar los registros de la tabla seleccionada al sistema de archivos HDFS, que en este caso es la tabla Phenotype.
  bin/sqoop import --connect "jdbc:oracle:thin:@localhost:1521:XE" --password "Bip4Cast" --username "USERPCB" --table "PCB.PHENOTYPE" --columns "ID_Patient,Birthdate,Sex,Diagnostic,Startdate,Sensitivetolithium, Sensitivetovalproic,Freqmaniccrises,Freqdepcrisis,Otherdx" --target-dir "/tmp/hdfs_sqoopimport" –verbose

- Podemos crear un *job Sqoop* y guardarlo para poder ejecutarlo posteriormente:
  bin/sqoop job --create import_job_pcb -- import –connect "jdbc:oracle:thin:@localhost:1521:XE" --password "Bip4Cast" --username "USERPCB" --table "PCB.PHENOTYPE" --columns

"ID_Patient,Birthdate,Sex,Diagnostic,Startdate,Sensitivetolithium, Sensitivetovalproic,Freqmaniccrises,Freqdepcrisis,Otherdx" --target-dir "/tmp/hdfs_sqoopimport" –verbose

- Para poder ejecutar el *job* guardado utilizamos la siguiente orden:
  bin/sqoop job --exec import_job_pcb

5.2.4. Implementación de estructuras para el análisis con Hive

La implementación de Hive nos permite tener un nivel de abstracción para realizar consultas sobre los datos obtenidos de los actígrafos utilizando HiveQL (lenguaje similar a SQL). Para mantener los archivos visibles dentro de HDFS todas las operaciones de creación de estructuras de tablas con Hive deberán incluir la sentencia EXTERNAL. A continuación, se muestra la creación, inserción y consulta de una tabla que tiene la misma estructura de los archivos recogidos desde los actígrafos:

- Creación de la tabla *hv_actigraph*

```
create external table hv_actigraph (
DateStamp timestamp,
X float,
Y float,
Z float,
Lux float,
Button char(1),
Temp decimal(2,1)
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
location '/userdata'
TBLPROPERTIES("skip.header.line.count"="100");
```

Hive utiliza *MapReduce* como su motor de ejecución lo que en algunos casos puede ocasionar que una consulta de datos lleve demasiado tiempo debido a las operaciones que realiza MapReduce sobre HDFS, pero existe una integración con la tecnología llamada *Tez* [68] que es un framework que optimiza la ejecución de tareas *MapReduce* todo coordinado por *YARN*, que mejora el rendimiento de Hive. En la Figura 5.8 se muestra las opciones del motor de ejecución de Hive: mr (MapReduce) y tez.

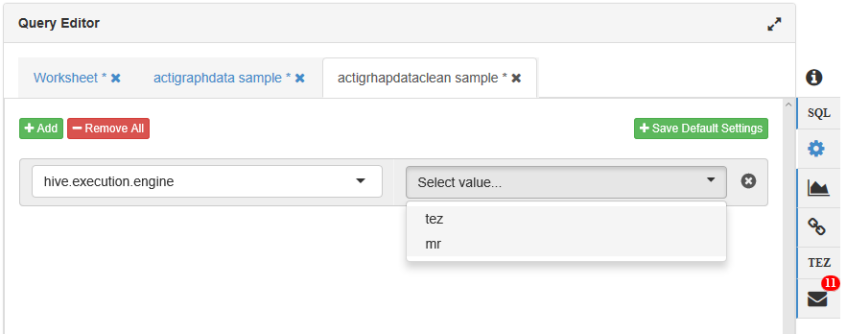


Figura 5.8. Motores de ejecución de Hive

5.2.5. Integración de R con Hadoop

Se ha realizado la implementación de R en el entorno Hadoop debido a que cuenta con librerías como GENEAread [69] que nos permite procesar los ficheros .bin obtenidos desde los actígrafos, la librearía rmr2 que permite construir operaciones MapReduce, la librería rhdfs que permite acceder a los ficheros almacenados en HDFS, además de que los profesionales involucrados en el proyecto cuentan con una alta experiencia en el manejo de la herramienta. En cuanto a la integración con Hadoop se han agregado los paquetes *rmr2* que permite la ejecución de algoritmos MapReduce y el paquete *rhdfs* que permite manipular ficheros del sistema de archivos de Hadoop (ver Apéndice: Instalación de R y RStudio en Hortonworks).

En la Figura 5.9 se aprecia la ejecución de una operación MapReduce realizada para probar la integración de R con Hadoop dentro de nuestra arquitectura.

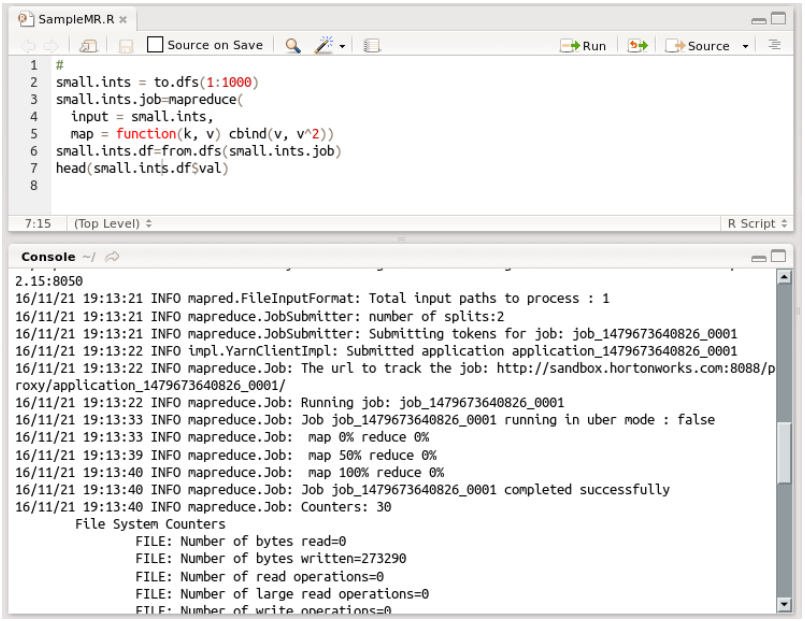


Figura 5.9. Ejecución de MapReduce en R integrado con Hortonworks

Para la lectura de los ficheros .bin del actígrafo se ha implementado un script que hace uso de la librería GENEAread, debido a que si se enviara el fichero sin convertirlo a .csv con la aplicación del actígrafo este se pueda leer para obtener los datos normalizados en un archivo .csv, cuyo código se muestra a continuación:

```
library(GENEAread)
library(mmap)
binfile = "gggg015-12-21 12-36-20.bin"
#Read in the entire file
procfile<-read.bin(binfile,downsample = 10)
print(procfile)
#Write CSV
write.csv(dataclean$data.out, file = dataclean$filename+".csv")
```

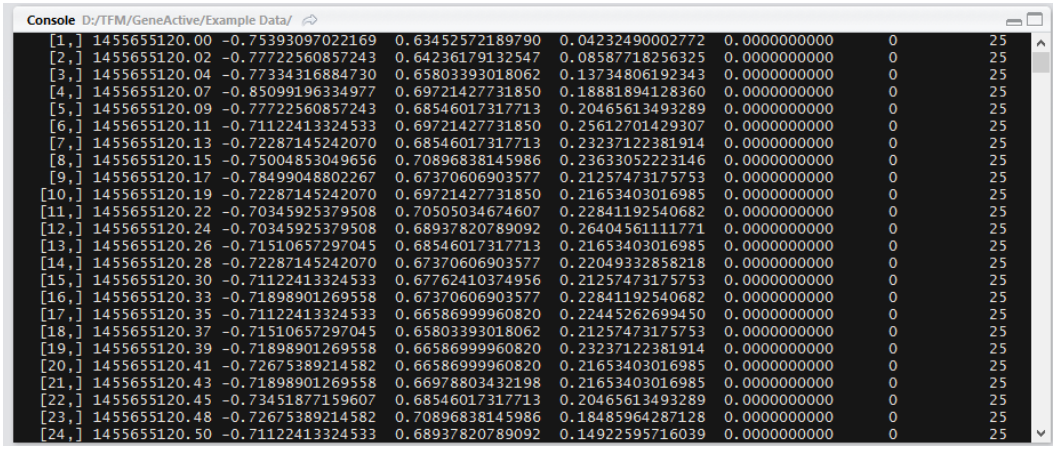


Figura 5.10. Listado de datos extraídos del actígrafo en R.

5.2.6. Integración de QlikView con Hive

La herramienta QlikView nos permite poder generar informes o interfaces donde se pueda visualizar los resultados del procesamiento de los datos, para su integración con la arquitectura se ha hecho uso del driver ODBC (*Open DataBase Connectivity*) para Hive desde el lado del usuario. Estos drivers se encuentran en la dirección *URL*: http://hortonworks.com/downloads/#addons.

En la Figura 5.11 se muestra la interfaz de configuración de la conexión con Hive desde un ordenador cliente.

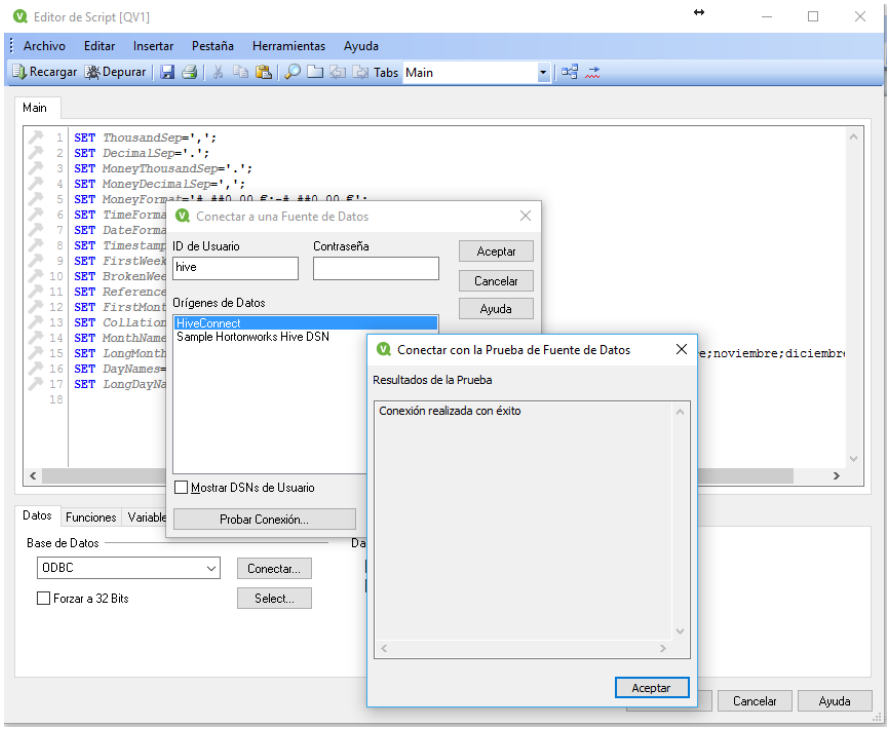


Figura 5.11. Configuración del driver ODBC para Hive

5.2.7. Implementación de seguridad con Ranger

En este servicio se han definido los roles de grupos y usuarios que harán uso del entorno Hadoop. Tomando como base la definición de roles de la arquitectura de referencia desarrollada en el capítulo 2, tenemos los siguientes grupos: Orquestación, Proveedor de Datos, Análisis y Consumidor de Datos. En la Figura 5.12 se muestra la interfaz de gestión de Ranger donde se tiene los servicios que soporta dentro de Hortonworks.

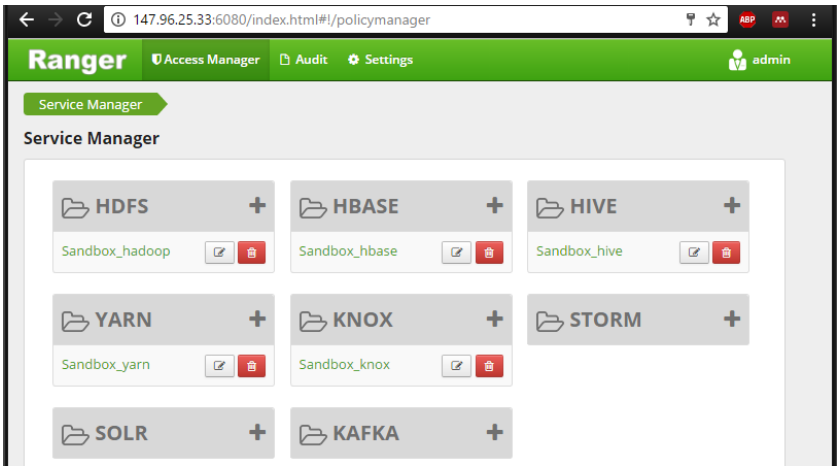


Figura 5.12. Interfaz de gestión de Ranger

En la Figura 5.13 se muestra la creación de una política de acceso en HDFS en la que se indica que el grupo “data_provider” con los permisos de lectura, escritura y ejecución, y el grupo “data_consumer” con los permisos de lectura y ejecución solo podrán acceder al directorio “/userdata” que es donde se encuentran los ficheros de los actígrafos.

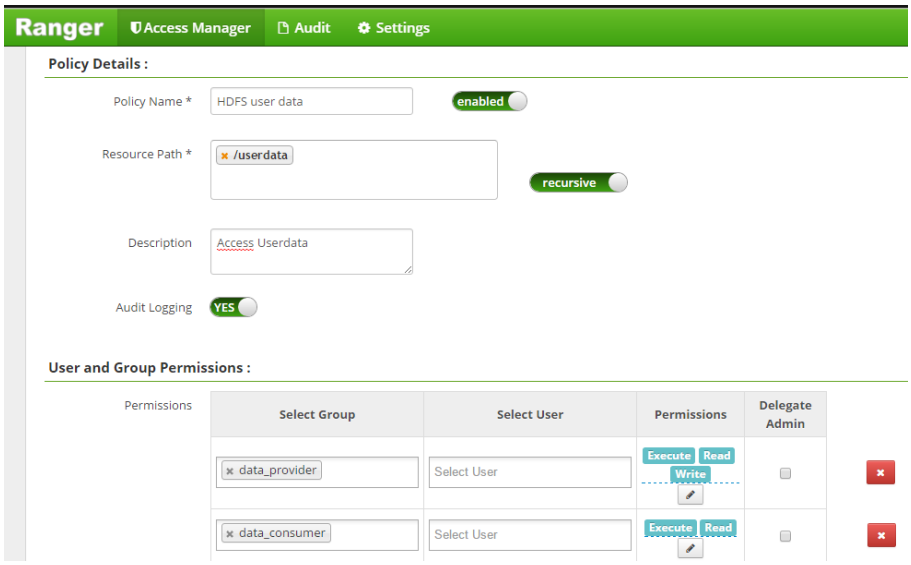


Figura 5.13. Configuración de política de acceso a HDFS

Otra política implementada es la de brindar acceso a las tablas que contienen los datos de los actígrafos a los grupos “data_provider” y “data_provider” con los permisos específicos de *select, index* y *create, index, select, update* respectivamente. Ver Figura 5.14.

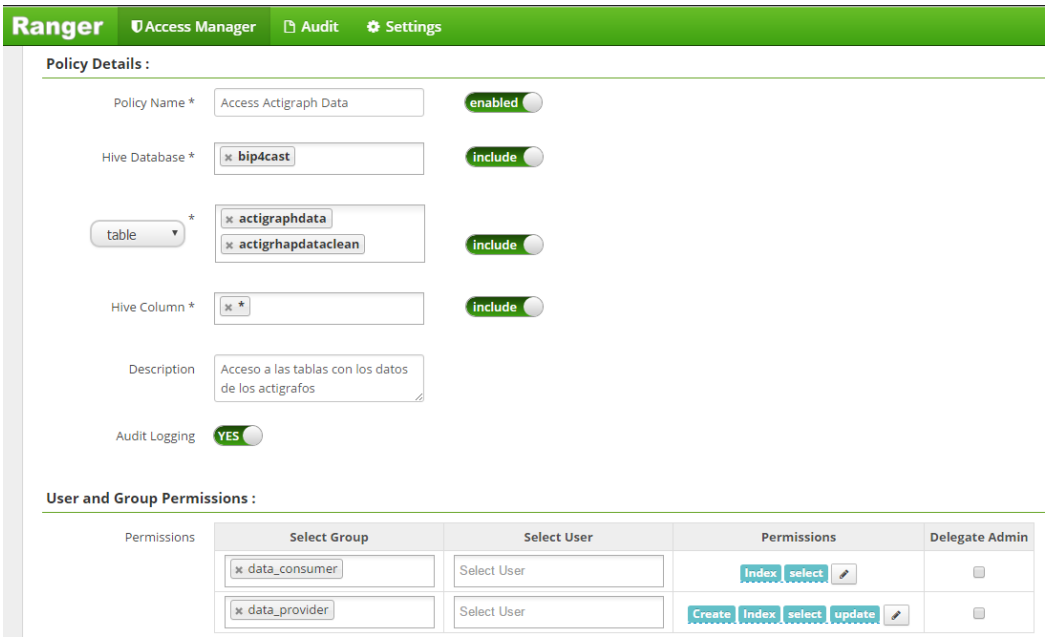


Figura 5.14. Configuración de política de acceso a HDFS

Una característica de Ranger es que además de permitir crear usuarios y grupos internos también es posible extraer los usuarios locales del sistema operativo, de un servicio LDAP o desde Directorio Activo.

5.3. Implementación de Hortonworks en la nube

Se ha realizado la implementación de Hortonworks en la nube por medio de la plataforma Azure de Microsoft para la que se ha utilizado una Suscripción gratuita de 30 días. La denominación de la versión de Hortonworks dentro de Azure es HDInsight, el cual fue descrito en la sección 3.3.2.

En la Figura 5.15 se muestra el panel de administración del clúster implementado dentro de Azure y en la Figura 5.16 se tiene la interfaz de gestión de Ambari.

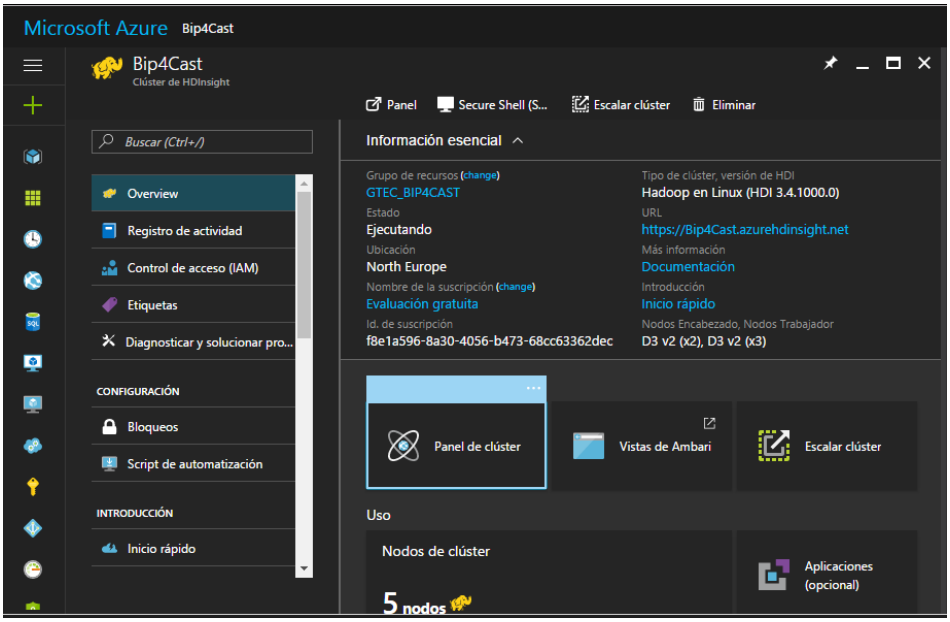


Figura 5.15. Interfaz del clúster HDInsight creado en Azure

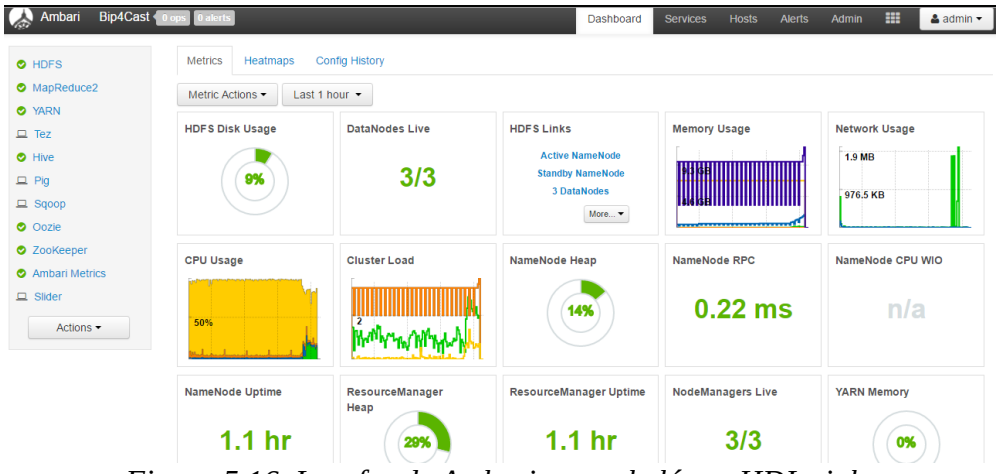


Figura 5.16. Interfaz de Ambari para el clúster HDInsight

Como se muestra en la Figuras 5.15 y 5.17 las características elegidas para el clúster de prueba son:

- Plataforma: Linux
- Número total de nodos: 5
- Nodos maestros: 2
- Nodos esclavos: 3
- Configuración por Nodo:
    - 14GB RAM
    - 4 núcleos
    - 8 discos hasta 1023GB c/u
    - 200GB de disco temporal
- Costo por nodo: 0.52 EUR / hora (2.6 EUR total)

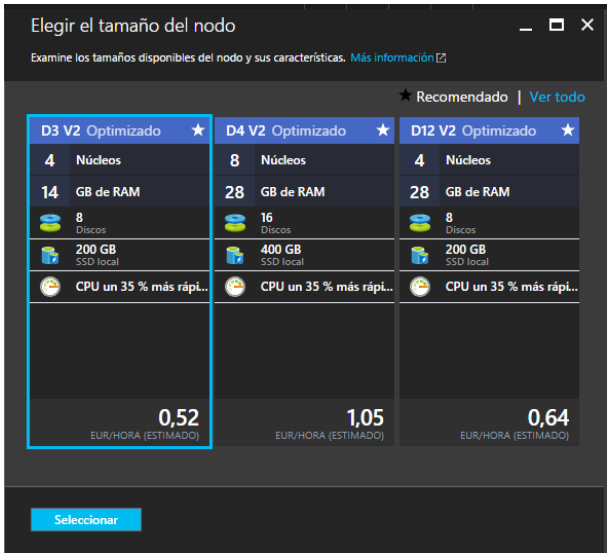


Figura 5.17. Características y costos de los nodos en Azure

Con las características configuradas podemos realizar los cálculos de los costos que conllevaría la implementación de una arquitectura, por ejemplo, si se quiere tener el servicio por seis meses y con una media de uso de 8 horas diarias se tiene:

(8 horas día x 2.6 EUR) x 180 días = 3744.00 EUR

Estos costos se deben evaluar al momento de la puesta en producción del proyecto ya que existen otros servicios ofertados de parte de empresas como Amazon y Google, por mencionar los más relevantes. En la Tabla 5.2 se muestra la comparativa de costos según las características de las instancias entre *Amazon*, *Azure* y *Google* que ha sido realizada por la empresa *RightScale*. Aquí se aprecia que *Google* tiene el precio más bajo para 3 escenarios y el más alto para 7 además tiene los costos más altos por GB de RAM. En cuanto a *Azure* este tiene el precio más bajo para 6

escenarios y el más alto para 4, y en comparación con Amazon tiene los precios más bajos en 7 de los 12 escenarios. Por último, Amazon tiene el precio más bajo para 3 escenarios y el más alto para 1 por lo suele ser la opción media entre los tres. Cabe indicar que estos precios se pueden ajustar dependiendo de los contratos que se tengan, por ejemplo, con Azure el contrato *Enterprise* o con Amazon las *Reserved Instances* los cuales tienen descuentos por pagos anuales.

Tabla 5.2. Comparativa de costos de instancias entre Amazon, Azure y Google

Resource Type (us-east, Linux)	AWS Instance	Azure Instance	Google Instance	AWS OD Hourly	Azure OD Hourly	Google OD Hourly	AWS /GB RAM	Azure /GB RAM	Google /GB RAM
Standard 2 vCPU w SSD	m3.large	D2 v2	n1-standard-2	$0.133	$0.114	$0.212	$0.017	$0.016	$0.028
Highmem 2 vCPU w SSD	r3.large	D11 v2	n1-highmem-2	$0.166	$0.149	$0.238	$0.011	$0.011	$0.018
Highcpu 2 vCPU w SSD	c3.large	F2	n1-highcpu-2	$0.105	$0.099	$0.188	$0.028	$0.025	$0.104
Standard 2 vCPU no SSD	m4.large	D2 v2	n1-standard-2	$0.108	$0.114	$0.100	$0.014	$0.016	$0.013
Highmem 2 vCPU no SSD	r4.large	D11 v2	n1-highmem-2	$0.133	$0.149	$0.126	$0.009	$0.011	$0.010
Highcpu 2 vCPU no SSD	c4.large	F2	n1-highcpu-2	$0.105	$0.099	$0.076	$0.027	$0.025	$0.042

En cuanto al almacenamiento se ha elegido el servicio de Almacenamiento de blobs [7] que permite almacenar datos binarios o de texto como documentos, archivos multimedia o un instalador de aplicación.

Cabe resaltar que durante la experimentación con el clúster no se ha encontrado opción para detener o pausar su ejecución, a lo que Azure recomienda eliminar el clúster ya que se seguirá facturando así no se haga uso del mismo y como los datos están almacenados en otro servicio estos no se perderán.

Capítulo 6. Experimentos y resultados

La evaluación de la arquitectura consiste en demostrar la funcionalidad y operatividad de la propuesta, enfocada en las fases de recolección o ingesta de datos e integración de las diversas fuentes. Además, se han realizado pruebas de consulta de datos y se han ejecutado algoritmos MapReduce que se describen en este capítulo.

6.1. Resultados obtenidos en la recolección de datos del actígrafo

En sus inicios el proyecto no contaba con una aplicación para enviar los ficheros al servidor por lo que se implementó dicha funcionalidad mediante SFTP lo que fue detallado en la sección 5.1. En la Tabla 6.1 se muestra la comparativa entre la operación inicial con versión original GENEActive y con la aplicación cliente SFTP.

Tabla 6.1. Comparativa entre la captura de datos con el GENEActiv original y con la aplicación cliente SFTP

Servicio	Modo de envío de datos	Servicio	Ventajas / Desventajas
GENEActiv PC Software original (local)	Extracción local, sin envío de datos.	Alternativas: Google Drive, Mega, etc.	- Pérdida de custodia de los datos, seguridad y tiempo. - Costos adicionales por el almacenamiento.
Aplicación cliente SFTP	Se mantiene la extracción local. Funcionalidad de envío mediante SFTP.	Servicio SFTP	- Se tienen los ficheros en el servidor para su tratamiento. - Se asegura la inmutabilidad de los datos. Recolección centralizada. - Se tiene una capa de seguridad. - No tienen costos adicionales.

Una vez implementada la funcionalidad de envío de ficheros se procedió a realizar diversas pruebas para determinar los tiempos de subida hacia el servidor, la Tabla 6.2 muestra la tasa de

transferencia de subida de los ficheros al servidor, además de los tiempos de extracción y envío por parte del usuario. Las pruebas se realizaron con una conexión de internet de 30Mbps de descarga y 6Mbps de subida (ADSL de casa).

Tabla 6.2. Tiempos y tasa de transferencia de los ficheros del actígarfo

Configuración del Actígrafo	Tamaño del fichero en bits	Tiempo de envío Tiempo Extracción + tiempo envío SFTP	Tasa de Transferencia Mbps = (Fichero (bits)/tiempo envío (seg))/1000000
Frecuencia = 40 Hz **Periodo = 18 días**	500 MB = 4194304000 bits	15 + 15 = 30 min	(4194304000/900)/1000000 = 4.66 Mbps
Frecuencia = 85.7 Hz **Periodo = 9 días**	800 MB = 6710886400 bits	20 + 20 = 40 min	(6710886400/1200)/1000000 = 5.59 Mbps
Frecuencia = 100 Hz **Periodo = 7 días**	900 MB = 7549747200 bits	25 + 25 = 50 min	(7549747200/1500)/1000000 = 5 Mbps

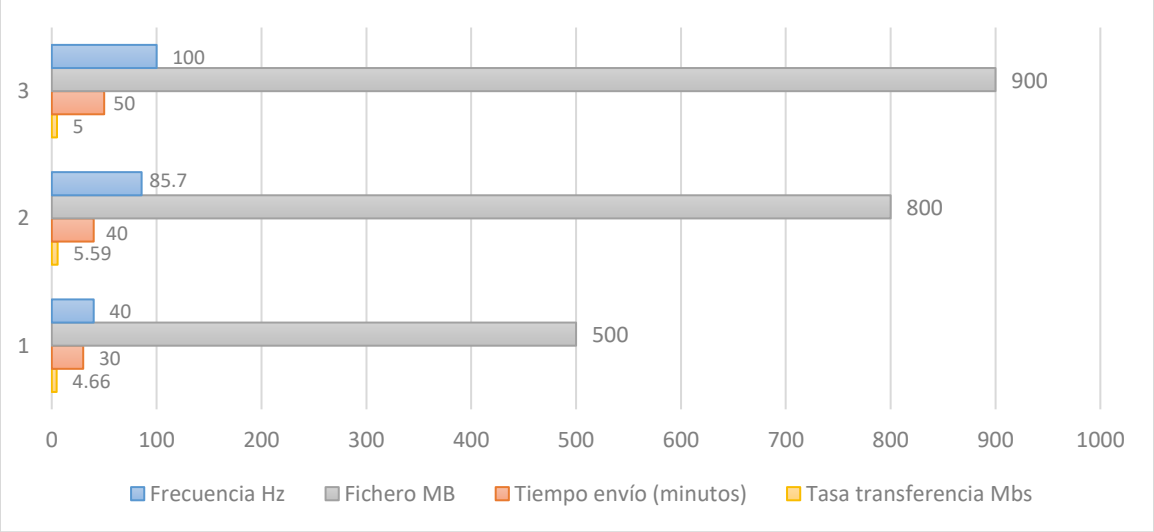


Figura 6.1. Tamaño, tiempo de envío y tasa de transferencia de los ficheros según la frecuencia de los actígrafos

De los resultados obtenidos en la medición de la tasa de transferencia para transmitir los ficheros podemos calcular, con la siguiente fórmula cual sería el ancho de banda necesario para el servicio SFTP, esto en función de la cantidad de usuarios concurrentes que tuviéramos.

*Ancho_Banda = Usuarios_concurrentes * Ancho_banda_usuario*

Se ha especificado que el proyecto contará con 25 pacientes en su primera etapa y estimando la media de la tasa de transferencia a 5 Mbps tendríamos lo siguiente:

*Ancho_Banda = 25 * 5 = 125Mbps*

Considerando que la Clínica Nuestra Señora de la Paz tiene contratado 25 Mbps simétricos, no podríamos atender la cantidad de usuarios concurrentes manteniendo tiempos de transferencia de la Tabla 6.2 por lo que estos cálculos nos dan una referencia de que ancho banda se necesitaría en un escenario determinado, dado que en nuestro caso transmitimos ficheros que son recolectados Después de días o semanas no podremos tener demasiados usuarios concurrentes o si se diera el caso se vería afectado el tiempo de transferencia.

La siguiente característica de la recolección de datos es el tamaño de los ficheros, lo que nos permitirá estimar la capacidad máxima de datos a ser recolectados en los discos con los que se cuenta. Se ha calculado que el tamaño de almacenamiento requerido con 85.7 Hz, durante 27 días y con 25 pacientes será de 58.6Gb.

$$EspacioGB_{27días} = (800MB * 3 * 25)/1024$$
$$EspacioGB_{27días} = 58.6GB$$

En el caso que se cuente con 1 disco de 1TB (solo para el servicio de SFTP) se podrá realizar el seguimiento de los 25 pacientes por un aproximado de 1 año: $((1024GB/58.6GB) * 27)/360) = 1.3$, lo que asegura el poder tener información relevante para realizar los análisis respectivos. Como los datos son almacenados en HDFS se debe considerar el factor de replicación que por defecto es 3 [70], por lo que se necesitará 3TB para HDFS.

La continua evaluación de los recursos con los que se cuenta nos permite ir enmarcando los parámetros del proyecto.

6.2. Resultados obtenidos en la implementación de Hadoop con Hortonworks

La implementación de la arquitectura propuesta con la distribución Hortonworks, muestra que las distribuciones al tener una variedad de aplicaciones integradas con Hadoop, y, sobre todo, pre

configuradas, minimiza los costos y tiempos de implementación. A continuación, se muestra la Tabla 6.3 con las valoraciones de determinadas características, resultado de la experiencia de implementación.

Tabla 6.3. Valoración de las características de implementación de Hortonworks

Característica	Valoración
Coste	Es código abierto
Sencillez	Dificultades encontradas en la configuración de los servicios.
Escalabilidad	Muy buena, sobre todo en el soporte de nuevos servicios.
Flexibilidad	Alta, por ser de código abierto
Investigación	Es una plataforma que invita a la experimentación.

6.2.1. Resultados obtenidos en la ingesta de datos

Una vez enviados los ficheros de los actígrafos al servidor estos son almacenados en el sistema de archivos de Hadoop (HDFS) con el fin de cumplir con los principios definidos en la sección 4.2.2. Debido a que en HDFS los ficheros serán distribuidos en los *Datanode* con un factor de replicación de cada bloque, esto nos asegura la integridad y validez de los datos, además de la inmutabilidad de los mismos.

Los archivos fueron recogidos por el servicio Flume para ser llevados a HDFS (ver Figuras 6.2 y 6.3). En un primer momento se hicieron las pruebas con la configuración por defecto en la asignación de memoria a la JVM lo que causaba el error: *OutOfMemoryError: Java heap space*, debido al tamaño de los ficheros que sobrepasaban la cantidad de memoria asignada, a lo que se procedió a configurar dicha propiedad en el fichero flume-env.sh: export JAVA_OPTS="-Xms512m -Xmx1024m -Dcom.sun.management.jmxremote", con lo que se dio solución al error y se pudo recolectar los ficheros.

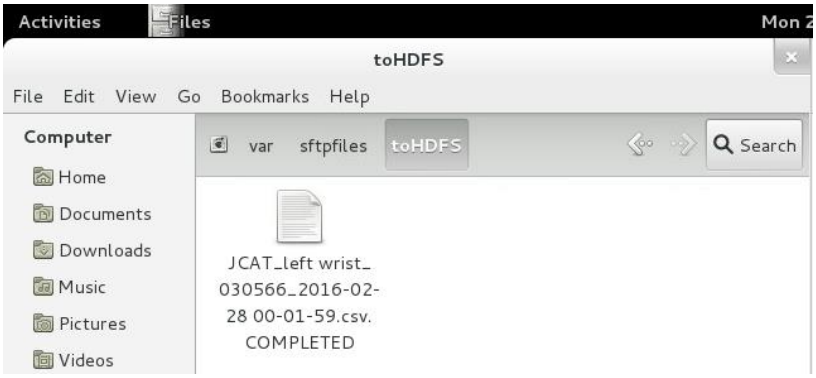
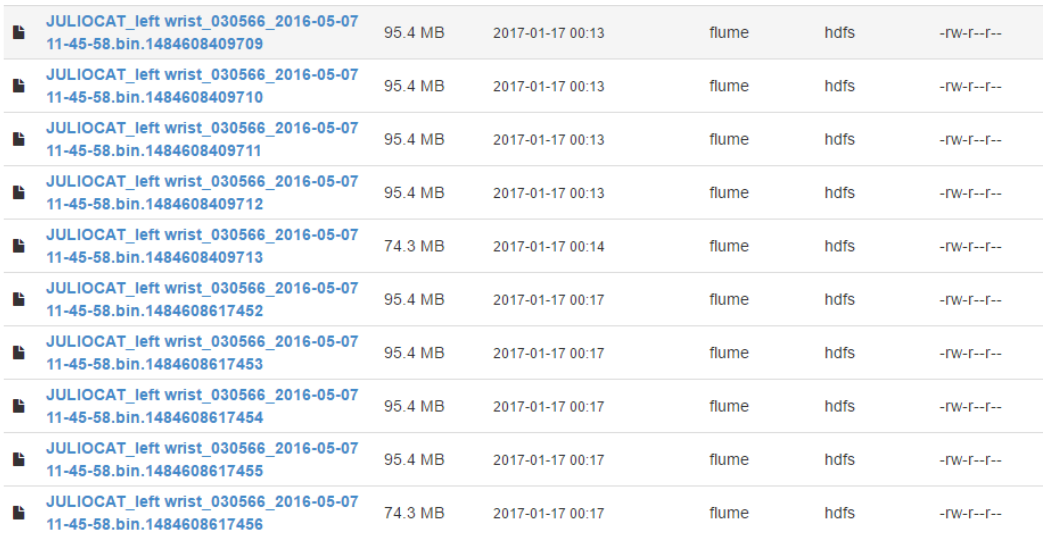


Figura 6.2. Fichero copiado por Flume a HDFS



JULIOCAT_left wrist_030566_2016-05-07 11-45-58.bin.1484608409709	95.4 MB	2017-01-17 00:13	flume	hdfs	-rw-r--r--
JULIOCAT_left wrist_030566_2016-05-07 11-45-58.bin.1484608409710	95.4 MB	2017-01-17 00:13	flume	hdfs	-rw-r--r--
JULIOCAT_left wrist_030566_2016-05-07 11-45-58.bin.1484608409711	95.4 MB	2017-01-17 00:13	flume	hdfs	-rw-r--r--
JULIOCAT_left wrist_030566_2016-05-07 11-45-58.bin.1484608409712	95.4 MB	2017-01-17 00:13	flume	hdfs	-rw-r--r--
JULIOCAT_left wrist_030566_2016-05-07 11-45-58.bin.1484608409713	74.3 MB	2017-01-17 00:14	flume	hdfs	-rw-r--r--
JULIOCAT_left wrist_030566_2016-05-07 11-45-58.bin.1484608617452	95.4 MB	2017-01-17 00:17	flume	hdfs	-rw-r--r--
JULIOCAT_left wrist_030566_2016-05-07 11-45-58.bin.1484608617453	95.4 MB	2017-01-17 00:17	flume	hdfs	-rw-r--r--
JULIOCAT_left wrist_030566_2016-05-07 11-45-58.bin.1484608617454	95.4 MB	2017-01-17 00:17	flume	hdfs	-rw-r--r--
JULIOCAT_left wrist_030566_2016-05-07 11-45-58.bin.1484608617455	95.4 MB	2017-01-17 00:17	flume	hdfs	-rw-r--r--
JULIOCAT_left wrist_030566_2016-05-07 11-45-58.bin.1484608617456	74.3 MB	2017-01-17 00:17	flume	hdfs	-rw-r--r--

Figura 6.3. Listado de ficheros en HDFS enviados por Flume

6.2.2. Resultados obtenidos en el análisis y consulta de datos

Con el fin de probar la arquitectura y como aporte de este trabajo se ha desarrollado un algoritmo MapReduce escrito en Python para el análisis de los ficheros recolectados por los actígrafos (ver Apéndice: Algoritmo MapReduce para el análisis de la actividad física de los pacientes).

Este algoritmo toma como referencia los cálculos mostrados en las macros para Excel implementadas por GENEActiv [71]. A continuación, se detallan los cálculos llevados al algoritmo MapReduce y los resultados que brinda.

- Bloque 1: Nos permite determinar los periodos de tiempo en los que el paciente ha tenido puesto o no el actígrafo para de este modo discriminar el cálculo de su actividad física. Los cálculos realizados son:

  Movimiento actual: $\sqrt{abs(x_1 - x_2 + y_1 - y_2 + z_1 - z_2)}$, para determinar la variación del movimiento entre periodos. Con respecto al fichero .csv los datos de x, y, z corresponden a las columnas 2, 3 y 4.

  Movimiento actual x SVM (suma de magnitudes vectoriales): Una vez obtenido el movimiento actual este se multiplica con el SVM (que indica si el actígrafo está en movimiento o no). El SVM es un dato que ya viene en el fichero csv y se encuentra en la columna 7.

  Media del Movimiento actual x SVM en el periodo de 1 hora, para determinar la continuidad de un comportamiento en un periodo de 1 hora.

  Valor mínimo de las medias en el periodo de 1 hora, con este cálculo final se puede inferir si el paciente tiene puesto o no el actígrafo.

- Bloque 2: Nos permite determinar la clasificación de la actividad física del paciente en 4 niveles, para esto se toma como referencia puntos de corte que indican el paso de un nivel a otro, estos puntos de corte pueden ser ajustados según se muestra en diversos estudios realizados en diferentes grupos de individuos [72]. Los puntos de corte hacen referencia al número de mediciones de un comportamiento en Cuentas por Minuto (CPM). A continuación, se detallan los valores utilizados para determinar estos niveles.

  Niveles de Actividad: Sedentario, Ligero, Moderado, Vigoroso.

  Puntos de Corte:

  - Sedentario a ligero: 410
  - Ligero a Moderado: 576
  - Moderado a Vigoroso: 1924

  Clasificación de los niveles: Para poder clasificar estos niveles se ha realizado la comparación de cada medición de SVM con los puntos de corte.

$$Si\ SVM > 1924 \rightarrow Vigoroso$$

$$Si\ 516 < SVM \leq 1924 \rightarrow Moderado$$

$$Si\ 410 < SVM \leq 516 \rightarrow Ligero$$

$$Si\ SVM \leq 416 \rightarrow Sedentario$$

Figura 6.4. Resultado del algoritmo MapReduce

Todos estos cálculos han sido llevados al algoritmo MapReduce implementado donde el Map extrae como clave la fecha y hora (sin minutos ni segundos) para poder agrupar los registros en periodos de 1 hora y el Reduce realiza los cálculos del Bloque 1 y 2 para generar los resultados. En la Figura 6.4 se muestra parte del resultado obtenido por el algoritmo donde se tiene la fecha y hora con un listado de 60 elementos que indican el nivel de actividad.

Otras pruebas que se han realizado son para verificar de la integración de R con Hadoop por medio de la librería *rmr2* [73] que permite ejecutar algoritmos MapReduce.

- Ejecución de MapReduce desde R en el clúster Hadoop.

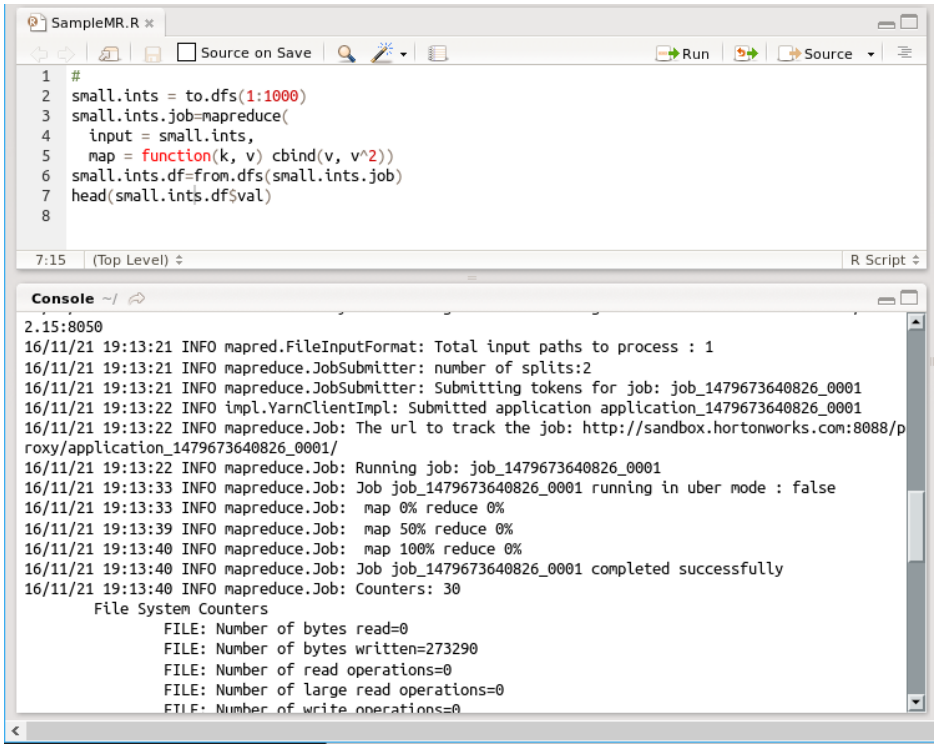


Figura 6.5. Ejecución de MapReduce en R

Capítulo 7. Conclusiones, trabajo futuro y publicaciones

7.1. Conclusiones

Como resultado de la presente investigación, se comprueba que las arquitecturas Big Data no siguen un estándar o referencia, al ser aún una tecnología en desarrollo. Recientemente se plantean referencias que buscan estandarizar los componentes mínimos necesarios con que debería contar una arquitectura. Estos marcos de referencia han sido considerados como puntos de partida para el diseño de la arquitectura Bip4Cast, dado que nos indican las funcionalidades y componentes genéricos con los que debe contar, y a partir de ellos se evaluaron las plataformas o aplicaciones que cumplen con los roles de referencia.

La implementación de la arquitectura propuesta con la distribución Hadoop Hortonworks nos indica que es posible utilizar plataformas de código abierto en entornos de producción sin que se tenga que sacrificar alguna característica o ventaja del ecosistema Hadoop, al contar con aplicaciones muy bien integradas y en constante actualización. Además, que en el caso de Hortonworks cuenta con socios estratégicos y forma parte de iniciativas de estandarización de implementaciones *open source* con Hadoop [74].

En el caso de nuestro prototipo implementado con la SandBox Hortonworks se ha configurado el uso de los recursos tanto por Flume, YARN y MapReduce lo que nos indica que a pesar de contar con una pre-configuración de Hadoop se hace necesaria la evaluación de los recursos con los que se cuenta para optimizar el funcionamiento de la arquitectura. Además, se han definido los requisitos mínimos de almacenamiento y ancho de banda con respecto a la recolección de datos desde los actígrafos, estas mediciones no incluyen otras fuentes de datos y que en caso de ser incluidas deben ser evaluadas para obtener los nuevos requerimientos de recursos. Para abordar las limitantes de los recursos con los que se cuenta se ha realizado una implementación de prueba de HDInsight en Microsoft Azure donde el problema del escalado tanto en almacenamiento como procesamiento queda cubierto en función de los costos que se puedan asumir y sobre todo si se cumplen con las disposiciones de protección de datos dentro de esta plataforma.

La recolección de datos de los actígrafos ha requerido integrar a la arquitectura el servicio SFTP para esto se ha implementado una aplicación cliente que permite tener los ficheros almacenados en un servidor central y la integración con Hadoop se ha implementado con Flume

que es quien vuelca los datos hacia HDFS de manera automática lo que demuestra que la arquitectura es flexible y escalable a la integración de diversas tecnologías.

Para la fase de análisis dentro de la arquitectura se ha implementado un algoritmo MapReduce con el cual se obtiene información relevante para el proyecto y se demuestra que los análisis de datos, que en principio se realizan en pequeñas cantidades de datos se pueden implementar en una arquitectura Big Data. Además, la integración de herramientas como R para poder utilizar sus librerías de manejo de los ficheros del actígrafo y construir operaciones MapReduce aprovechando las funciones de análisis con las que cuenta el lenguaje nos indica que la posibilidad de integración de tecnologías comúnmente usadas permite volcar la experiencia y conocimientos adquiridos hacia proyectos Big Data.

Desde el punto de vista de la arquitectura, después de las pruebas y la experimentación con las herramientas que integran la distribución Hadoop se puede llegar a la conclusión que todas son susceptibles a mejoras, sea en su rendimiento, en su facilidad de instalación o configuración. De igual manera, en el caso del Actígrafo, este puede ser actualizado a uno que envíe los datos al servidor en tiempo real lo que ocasionará que la arquitectura tenga que soportar el nuevo tipo de recolección de datos.

7.2. Trabajo futuro

Uno de los puntos críticos a tomar en cuenta es el escalado de la arquitectura sobre todo en el almacenamiento. Con 1TB en el servidor de desarrollo y pruebas, y con una media de 58.6GB mensuales con 25 pacientes, la recolección de datos puede limitarse a poco más de un año. Por esto, se debe realizar el análisis de los servicios de almacenamiento en la nube frente a los costos de adquisición de una infraestructura de mayor capacidad. No obstante, si se decide utilizar los servicios de la nube se debe tener en claro las políticas de seguridad, privacidad y tratamiento de datos de estos servicios, ya que en nuestro caso al ser datos médicos altamente sensibles se podría incurrir en alguna infracción.

En el caso de que se quiera replicar la arquitectura montada en Hortonworks en más hospitales esta tendría que volver a configurarse según los recursos con los que se cuentan, abriéndose una línea de investigación a como la arquitectura pudiera evaluar los recursos del nuevo sitio para reconfigurar sus servicios y aplicaciones. Además, quedaría la tarea de dotar a la

arquitectura de portabilidad sin importar el sistema operativo esto se lograría con tecnologías como *Docker*.

En cuanto a la recolección de datos dentro de la arquitectura planteada se tiene configurado el volcado de los ficheros hacia HDFS, pero esto podría cambiar si se integran actígrafos con envío de datos en tiempo real requiriéndose la implementación o configuración de nuevos servicios.

Por último, debido a que se tienen más fuentes de datos por integrar al proyecto como imágenes, audio o vídeo que además son datos sensibles de los pacientes queda implementar procesos de anonimización y codificación que pueden producir *overhead* dentro de la arquitectura y en consecuencia afectar el escalado al requerir mayores recursos.

7.3. Publicaciones Relacionadas

Como consecuencia de este trabajo se han realizado las siguientes publicaciones:

V. López, G. Valverde, J. C. Anchiraico y D. Urgelés. "Specification of a Cad Prediction System for Bipolar Disorder", Uncertainty Modelling in Knowledge Engineering and Decision Making, Proceedings of the 12th International FLINS conference, World Scientific Proceedings Series on Computer Engineering and Information Science, vol. 10, pp. 162-167. ISBN 978-981-3146-96-9, World Scientific, 2016. [Core B]

V. López, D. Urgelés, V. Mariscal y J. C. Anchiraico. "Integración de datos masivos como input de análisis predictivos: Trastornos de bipolaridad y drogodependencia". Póster presentado al XXXVI Congreso Nacional Estadística e Investigación Operativa, SEIO, 12 septiembre de 2016, Toledo, España.

V. López, G. Miñana, B. Gonzáles, O. Sánchez, G. Valverde, J. C. Anchiraico y D. Urgelés. "GRASIA-GTeC: Big Data en salud y ciencias sociales". Póster presentado a la Jornada FuzzyMAD, 16 de diciembre de 2016, Madrid, España.

Chapter 7. Conclusions, future work and publications

7.1. Conclusions

As a result of the present investigation, it is verified that the Big Data architectures do not follow a standard or reference, being still a technology in development. Recently, references have been made that seek to standardize the minimum necessary components with which an architecture should count. These frames of reference have been considered as starting points for the design of the Bip4Cast architecture, given that they indicate the generic functionalities and components with which they must count, and from them we evaluated the platforms or applications that fulfill the roles of reference.

The implementation of the proposed architecture with the Hadoop Hortonworks distribution tells us that it is possible to use open source platforms in production environments without having to sacrifice some feature or advantage of the Hadoop ecosystem, having very well integrated and constantly updated applications. In addition, in the case of Hortonworks has strategic partners and is part of standardization initiatives of open source implementations with Hadoop [74].

In the case of our prototype implemented with the Sandbox Hortonworks has been configured the use of resources by Flume, YARN and MapReduce which tells us that despite having a pre-configuration of Hadoop makes it necessary to evaluate the resources to optimize the operation of the architecture. In addition, the minimum storage and bandwidth requirements have been defined with respect to data collection from accelerometers, these measurements do not include other data sources and that, if included, must be evaluated to obtain the new resource requirements. In order to address the limitations of the available resources, an HDInsight test implementation has been implemented in Microsoft Azure where the problem of scaling in both storage and processing is covered in terms of the costs that can be assumed and especially if Comply with data protection provisions within this platform.

The data collection of the accelerometers has required integration of the SFTP service for this has been implemented a client application that allows to have the files stored in a central server and the integration with Hadoop has been implemented with Flume that is who transfer the data To HDFS automatically demonstrating that the architecture is flexible and scalable to the integration of various technologies.

For the analysis phase within the architecture, a MapReduce algorithm has been implemented which provides information relevant to the project and it is demonstrated that data analyzes, which are done in small amounts of data, can be implemented in an architecture Big Data. In addition, the integration of tools such as R to be able to use their libraries to manage the files of the accelerometer and to construct MapReduce operations taking advantage of the analysis functions with which the language tells us that the possibility of integration of commonly used technologies allows to bring the experience and knowledge gained towards Big Data projects.

From the architectural point of view, after testing and experimenting with the tools that make up the Hadoop distribution one can conclude that all are susceptible to improvements, be it performance, ease of installation or configuration. Similarly, in the case of the accelerometer, this can be updated to one that sends the data to the server in real time which will cause the architecture to have to support the new type of data collection.

7.2. Future work

One of the critical points is the scaling of architecture, especially in storage. With 1TB on the testing server and with an average of 58.6GB per month with 25 patients, data collection can be limited to just over a year. For this reason, the analysis of storage services in the cloud should be performed against the costs of acquiring a higher capacity infrastructure. However, if it is decided to use the cloud services, should be clear about the security, privacy and data processing policies of these services, since in our case, being highly sensitive medical data could incur some infringement.

In case is needed to replicate the architecture assembled in Hortonworks in more hospitals this would have to be reconfigured according to the resources that are counted, opening a line of research to how the architecture could evaluate the resources of the new site to reconfigure its services and applications. In addition, the task of giving the architecture portability regardless of the operating system would be achieved with technologies such as Docker.

As far as data collection is concerned, the file transfer is configured to HDFS, but this could change if data collection is integrated with real-time data transmission, requiring the implementation or configuration of new services.

Finally, because there are more data sources to integrate into the project such as images, audio or video that are also sensitive data of patients, it is necessary to implement anonymization

and coding processes that can produce overhead within the architecture and, consequently, affect the scalability by needing more resources.

7.3. Related Publications

As a result of this work the following publications have been made:

V. López, G. Valverde, J. C. Anchiraico y D. Urgelés. “Specification of a Cad Prediction System for Bipolar Disorder”, Uncertainty Modelling in Knowledge Engineering and Decision Making, Proceedings of the 12th International FLINS conference, World Scientific Proceedings Series on Computer Engineering and Information Science, vol. 10, pp. 162-167. ISBN 978-981-3146-96-9, World Scientific, 2016. [Core B]

V. López, D. Urgelés, V. Mariscal and J. C. Anchiraico (2016, September). "Integration of mass data as input of predictive analysis: Bipolar disorder and drug addiction". Poster presented at the XXXVI National Statistical Congress and Operational Research, SEIO, September 12, 2016, Toledo, Spain.

V. López, G. Miñana, B. Gonzales, O. Sánchez, G. Valverde, J. C. Anchiraico and D. Urgelés (2016, December). "GRASIA-GTeC: Big Data in health and social sciences". Poster presented at the FuzzyMAD Day, December 16, 2016, Madrid, Spain.

Bibliografía

[1] K. R. Merikangas, R. Jin, J.-P. He, R. C. Kessler, S. Lee, N. A. Sampson, M. C. Viana, L. H. Andrade, C. Hu, E. G. Karam, M. Ladea, M. E. Medina-Mora, Y. Ono, J. Posada-Villa, R. Sagar, J. E. Wells, and Z. Zarkov, "Prevalence and correlates of bipolar spectrum disorder in the world mental health survey initiative.," *Arch. Gen. Psychiatry*, vol. 68, no. 3, pp. 241–51, Mar. 2011.

[2] S. Harvey, "Bipolar disorder," *University of Maryland Medical Center*, 2013. [Online]. Available: https://www.nimh.nih.gov/health/topics/bipolar-disorder/index.shtml. [Accessed: 15-Jun-2016].

[3] K. Hawton, L. Sutton, C. Haw, J. Sinclair, and L. Harriss, "Suicide and attempted suicide in bipolar disorder: A systematic review of risk factors," *J. Clin. Psychiatry*, vol. 66, no. 6, pp. 693–704, Jun. 2005.

[4] J. L. Martin and A. D. Hakim, "Wrist actigraphy," *Chest*, vol. 139, no. 6. American College of Chest Physicians, pp. 1514–1527, Jun-2011.

[5] N. Big Data Public Working Group, "NIST Big Data Interoperability Framework: Volume 6, Reference Architecture."

[6] "Apache™ Hadoop®." [Online]. Available: http://hadoop.apache.org/.

[7] "Uso del almacenamiento de blobs de Azure compatibles con HDFS con Hadoop en HDInsight." [Online]. Available: https://azure.microsoft.com/es-es/documentation/articles/hdinsight-hadoop-use-blob-storage/.

[8] R. Kumar, B. B. Parashar, S. Gupta, Y. Sharma, and N. Gupta, "Apache Hadoop , NoSQL and NewSQL Solutions of Big Data," *Int. J. Adv. Found. Res. Sci. Eng.*, vol. 1, no. 6, 2014.

[9] W. Read, T. Report, and K. Takeaways, "The Forrester Wave™: Big Data Hadoop Distributions, Q1 2016," 2016.

[10] "Hortonworks: Open and Connected Data Platforms." [Online]. Available: http://hortonworks.com/.

[11] C. Tan, L. Sun, and K. Liu, "Association for Information Systems AIS Electronic Library (AISeL) Big Data Architecture for Pervasive Healthcare: A Literature Review," 2015.

[12] N. Marz and J. (James O. . Warren, *Big data : principles and best practices of scalable*

real-time data systems. 2015.

[13] X. Chen, S. Wang, Y. Dong, and X. Wang, "Big Data Storage Architecture Design in Cloud Computing," Springer Singapore, 2016, pp. 7–14.

[14] Y. Demchenko, C. De Laat, and P. Membrey, "Defining architecture components of the Big Data Ecosystem," in *2014 International Conference on Collaboration Technologies and Systems, CTS 2014*, 2014, pp. 104–112.

[15] "National Institute of Standards and Technology." [Online]. Available: http://www.nist.gov/.

[16] V. T. Van Hees, S. Sabia, K. N. Anderson, S. J. Denton, J. Oliver, M. Catt, J. G. Abell, M. Kivimäki, M. I. Trenell, and A. Singh-Manoux, "A novel, open access method to assess sleep duration using a wrist-worn accelerometer," *PLoS One*, vol. 10, no. 11, p. e0142533, Nov. 2015.

[17] M. P. Buman, F. Hu, E. Newman, A. F. Smeaton, and D. R. Epstein, "Behavioral Periodicity Detection from 24 h Wrist Accelerometry and Associations with Cardiometabolic Risk and Health-Related Quality of Life," *Biomed Res. Int.*, vol. 2016, 2016.

[18] GENEActiv, "GENEActiv Instructions." [Online]. Available: http://www.geneactiv.org/wp-content/uploads/2014/03/geneactiv_instruction_manual_v1.2.pdf. [Accessed: 28-Apr-2016].

[19] P. Pekka Pääkkönen and D. Pakkala, "Reference Architecture and Classification of Technologies, Products and Services for Big Data Systems," *Big Data Research*, vol. 2, no. 4. pp. 166–186, 2015.

[20] E. Dumbill, "What is Big Data?," in *Planning for Big Data*, 1st ed., O'Reilly Media, Ed. Amazon Media EU, 2012, p. 84.

[21] E. Martín, "Apuntes y Transparencias de la asignatura Sistemas de Gestión de Datos y de la Información." .

[22] N. Big Data Public Working Group, "NIST Big Data Interoperability Framework: Volume 1, Definitions," *NIST Spec. Publ.*, pp. 1500–1.

[23] Oracle, "An Enterprise Architect's Guide to Big Data Reference Architecture Overview," 2016.

[24] N. Big Data Public Working Group, “NIST Big Data Interoperability Framework: Volume 3, Use Cases and General Requirements,” *NIST Spec. Publ.*, pp. 1500–3.

[25] G. Miller and P. Mork, “From Data to Decisions: A Value Chain for Big Data.”

[26] Y. Zhao, E. A. Lee, R. Advisor, and A. Sangiovanni-Vincentelli, “A Model of Computation with Push and Pull Processing,” 2003.

[27] J. Hurwitz, *Big data for dummies*. John Wiley & Sons, 2013.

[28] D. Keim, H. Qu, and K.-L. Ma, “Big-Data Visualization,” *IEEE Comput. Graph. Appl.*, vol. 33, no. 4, pp. 20–21, Jul. 2013.

[29] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters.”

[30] “BSP Computing.” [Online]. Available: http://web.archive.org/web/20000826223421/web.comlab.ox.ac.uk/oucl/research/highlights/bsp_computing.html. [Accessed: 26-Aug-2016].

[31] P. J. Sadalage and M. Fowler, *NoSQL distilled : a brief guide to the emerging world of polyglot persistence*. Addison-Wesley, 2013.

[32] Han Hu, Yonggang Wen, Tat-Seng Chua, and Xuelong Li, “Toward Scalable Systems for Big Data Analytics: A Technology Tutorial,” *IEEE Access*, vol. 2, pp. 652–687, 2014.

[33] “Introduction to Real time Big Data processing a... | CA Communities.” [Online]. Available: https://communities.ca.com/community/ca-apm/blog/2016/03/03/real-time-big-data-processing-architectures-lambda-kappa-and-zeta.

[34] V. K. Vavilapalli, S. Seth, B. Saha, C. Curino, O. O’Malley, S. Radia, B. Reed, E. Baldeschwieler, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, and H. Shah, “Apache Hadoop YARN,” in *Proceedings of the 4th annual Symposium on Cloud Computing - SOCC ’13*, 2013, pp. 1–16.

[35] “Apache Ambari.” [Online]. Available: http://ambari.apache.org/.

[36] “Apache ZooKeeper.” [Online]. Available: http://zookeeper.apache.org/.

[37] “IBM - What is HBase?” [Online]. Available: https://www-01.ibm.com/software/data/infosphere/hadoop/hbase/.

[38] “IBM - What is Flume.” [Online]. Available: https://www-01.ibm.com/software/data/infosphere/hadoop/flume/.

[39] “Apache Flume.” [Online]. Available: https://flume.apache.org/.

[40] “Apache Sqoop.” [Online]. Available:

https://blogs.apache.org/sqoop/entry/apache_sqoop_graduates_from_incubator.

[41] “HadoopMapReduce - Hadoop Wiki.” [Online]. Available: http://wiki.apache.org/hadoop/HadoopMapReduce.

[42] “QlikView - Hortonworks.” [Online]. Available: http://hortonworks.com/hadoop-tutorial/business-discovery-and-visualizing-your-data-in-hdp-using-qlikview/. [Accessed: 02-Nov-2016].

[43] B. Azarmi, *Scalable Big Data Architecture: A practitioners guide to choosing relevant Big Data architecture*. 2015.

[44] “MapR.” [Online]. Available: https://www.mapr.com.

[45] “Amazon EMR.” [Online]. Available: http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-what-is-emr.html.

[46] “Introducción a HDInsight - Microsoft Azure.” [Online]. Available: https://azure.microsoft.com/es-es/documentation/articles/hdinsight-hadoop-introduction/.

[47] B. Vorhies, “Cloudera vs Hortonworks vs MapR – Has MapR Won This Contest Already?” [Online]. Available: http://data-magnum.com/cloudera-vs-hortonworks-vs-mapr-has-mapr-already-won-this-contest/.

[48] “The Forrester Wave™: Big Data Hadoop Cloud Solutions, Q2 2016.” [Online]. Available: http://reprints.forrester.com/#/assets/2/108/’RES126541’/reports. [Accessed: 30-Oct-2016].

[49] Gobierno de España, “Ley Orgánica 15/1999, de Protección de Datos de Carácter Personal.”

[50] “La ciclación rápida: características clínicas y tratamiento.” [Online]. Available: http://www.forumclinic.org/es/trastorno-bipolar/noticias/la-ciclación-rápida-características-clínicas-y-tratamiento. [Accessed: 27-Nov-2016].

[51] “GENEActiv Original - Wrist-Worn Actigraphy Device | GENEActiv Accelerometers.” [Online]. Available: http://www.geneactiv.org/actigraphy/geneactiv-original/.

[52] “Production Notes — MongoDB Manual 3.2.” [Online]. Available: https://docs.mongodb.com/manual/administration/production-notes/.

[53] “FTPS (FTP over SSL) vs. SFTP (SSH File Transfer Protocol): what to choose.” [Online]. Available: https://www.eldos.com/security/articles/4672.php?page=all#. [Accessed: 05-

May-2016].

[54] A. D. Christina Dillon, Cormac Powell, Kieran Dowd, Brian Carson, “Criterion validity and calibration of the GENEActiv accelerometer in adults,” 2015.

[55] J.-H. Lee, H.-J. Park, and Y.-N. Kim, “Monitoring obstructive sleep apnea with electrocardiography and 3-axis acceleration sensor,” p. 74, Jul. 2015.

[56] C. M. Duan, “Design of Big Data Processing System Architecture Based on Hadoop under the Cloud Computing,” *Appl. Mech. Mater.*, vol. 556–562, pp. 6302–6306, May 2014.

[57] B. Efron and R. Tibshirani, *An introduction to the bootstrap*. Chapman & Hall, 1994.

[58] E. Sevilla Rivera and D. S. Vargas, “Diseño de una arquitectura para Big Data.”

[59] “Qt Designer Manual.” [Online]. Available: http://doc.qt.io/qt-5/qtdesigner-manual.html. [Accessed: 30-May-2016].

[60] “PyQt.” [Online]. Available: https://www.riverbankcomputing.com/software/pyqt/intro. [Accessed: 04-Jun-2016].

[61] “pysftp 0.2.9.” [Online]. Available: https://pypi.python.org/pypi/pysftp. [Accessed: 01-Jun-2016].

[62] “cx_Freeze 5.0.1 documentation.” [Online]. Available: http://cx-freeze.readthedocs.io/en/latest/distutils.html. [Accessed: 29-Nov-2016].

[63] “SharpSSH - A Secure Shell (SSH) library for .NET.” [Online]. Available: http://www.tamirgal.com/blog/page/sharpssh.aspx. [Accessed: 25-Oct-2016].

[64] “Hortonworks Documentation.” [Online]. Available: http://docs.hortonworks.com/index.html. [Accessed: 02-Dec-2016].

[65] “Understanding Memory Management - JVM Heap.” [Online]. Available: https://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/garbage_collect.html. [Accessed: 05-Dec-2016].

[66] “Determine HDP Memory Configuration Settings - Hortonworks Data Platform.” [Online]. Available: https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.4.2/bk_installing_manually_book/content/determine-hdp-memory-config.html. [Accessed: 03-Dec-2016].

[67] C. Wang, I. A. Rayan, and K. Schwan, “Faster, Larger, Easier: Reining Real-Time Big Data Processing in Cloud,” 2012.

[68] “Apache Tez - Hortonworks.” [Online]. Available: http://hortonworks.com/apache/tez/.

[Accessed: 28-Nov-2016].

[69] Z. Fang, “Package ‘GENEAread’ For Reading Binary files,” 2015. [Online]. Available: https://cran.r-project.org/web/packages/GENEAread/GENEAread.pdf. [Accessed: 31-May-2016].

[70] “Hadoop Cluster Sizing | Distributed Systems Architecture.” [Online]. Available: https://0x0fff.com/hadoop-cluster-sizing/. [Accessed: 24-Jan-2017].

[71] “GENEActiv | Open Platform – Macros.” [Online]. Available: https://open.geneactiv.org/geneactiv_macros.html. [Accessed: 26-Dec-2016].

[72] “ActiGraph Cut Points.” [Online]. Available: https://actigraph.desk.com/customer/en/portal/articles/2515803-what-s-the-difference-among-the-cut-points-available-in-actilife-. [Accessed: 05-Nov-2016].

[73] RevolutionAnalytics, “RHadoop,” 2015. [Online]. Available: https://github.com/RevolutionAnalytics/RHadoop.

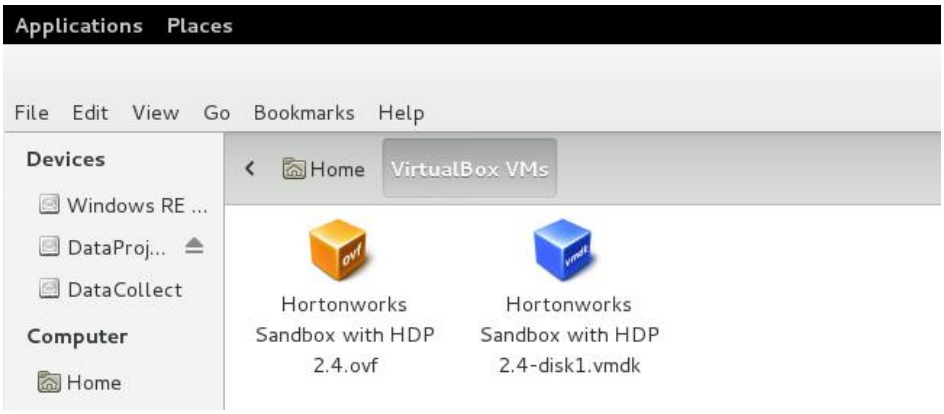
[74] “Specifications - ODPi.” [Online]. Available: https://www.odpi.org/specifications.

Apéndice: Instalación de la SandBox Hortonworks

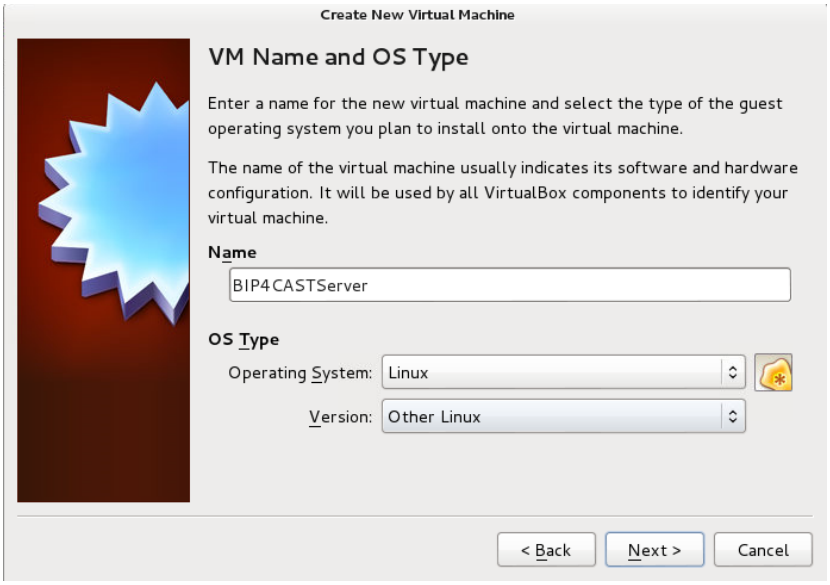
Requerimientos de la SandBox:

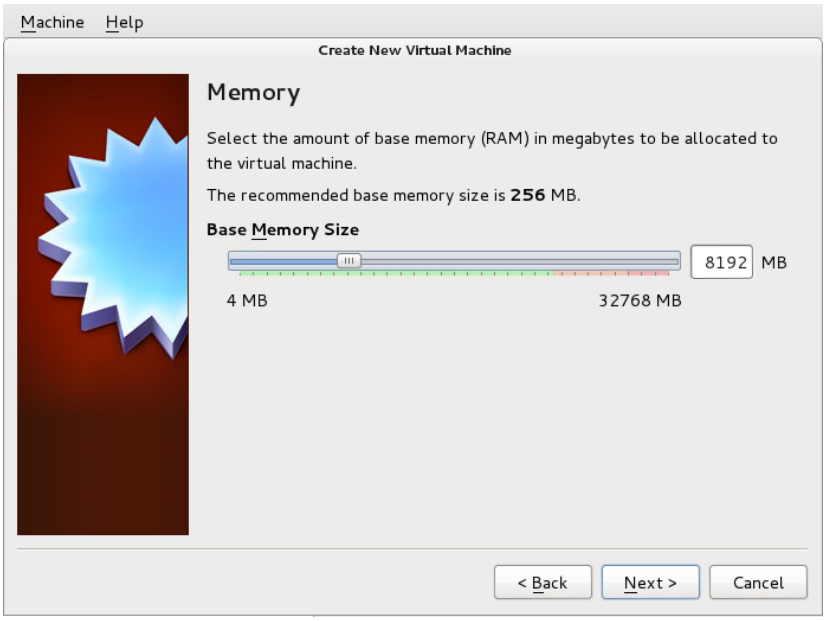
- Memoria: 8GB
- Procesadores: 4
- Disco Duro: 20 GB – 50GB
- Network Adapter: NAT

Las siguientes figuras muestran los pasos seguidos para la implementación de Hortonworks, donde se han tenido dificultades en la importación del archivo *OVA*, por lo que se ha encontrado una solución extrayendo los archivos contenidos dentro del mismo: *ovf* y *vmdk*.

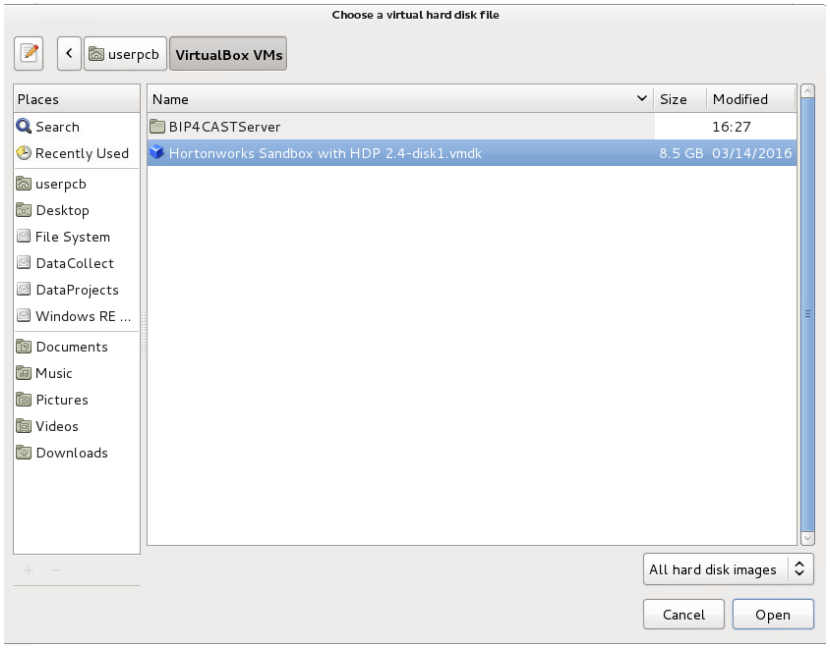


Una vez extraídos los ficheros creamos una nueva máquina virtual especificando los requisitos mínimos de la SandBox, por ejemplo, 8GB de RAM.

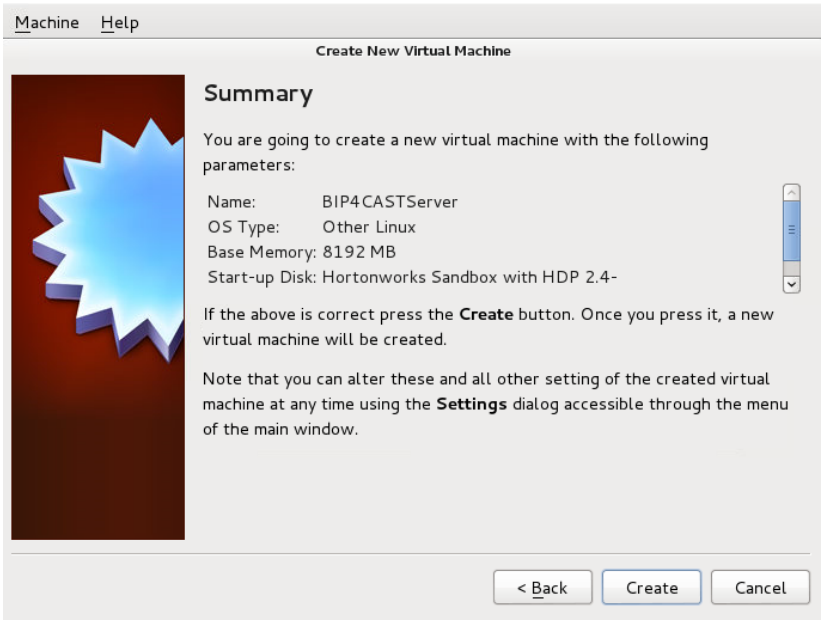




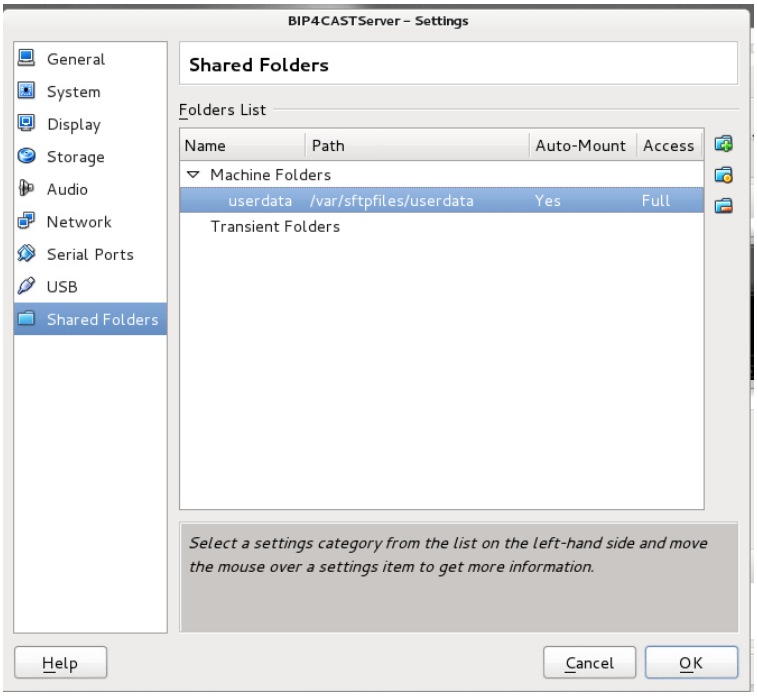
Cuando lleguemos al apartado de Disco Duro Virtual, seleccionamos “Usar un disco duro virtual existente”, para luego elegir el archivo de extensión *vmdk* que se ha extraído anteriormente. Es en este archivo donde se encuentra toda la instalación de la distribución Hortonworks.



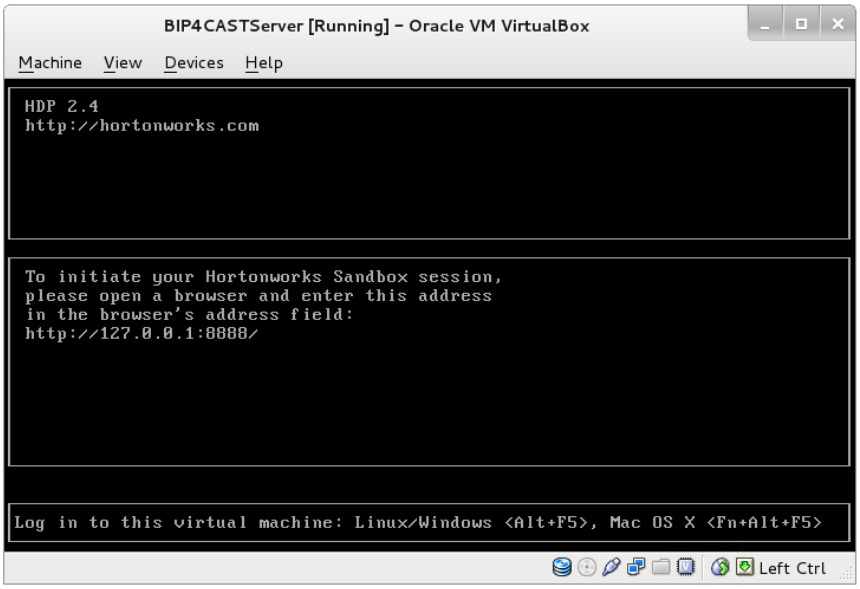
Y por último creamos la máquina virtual.



Dejamos configurado el acceso al directorio del servidor donde se almacenan los ficheros con los datos de los actígrafos. Para esto, agregamos este directorio en la opción “Directorios compartidos” de la máquina virtual. Esto para que Flume pueda acceder a los ficheros y poderlos enviar al sistema de archivos HDFS.



Ahora ya podemos dar inicio a la máquina virtual.



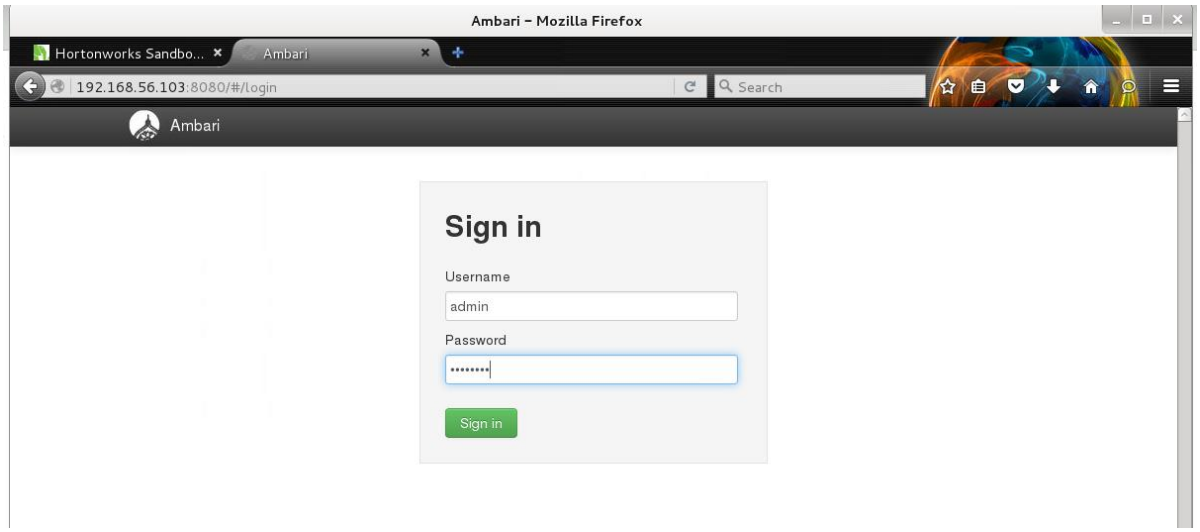
Para poder acceder a los servicios que provee la SandBox debemos tener en cuenta las direcciones URL y los puertos donde se han configurado dichos servicios. Estos puertos se registran en la opción de "Reenvío de puertos" para que se puedan acceder desde el host y en caso de contar con una IP pública tener acceso por medio de internet. Algunos puertos de uso común en el host se han cambiado para no generar conflictos, por ejemplo, el puerto que utiliza SSH es el 22 por lo que se redirecciona al 2222 en el host.

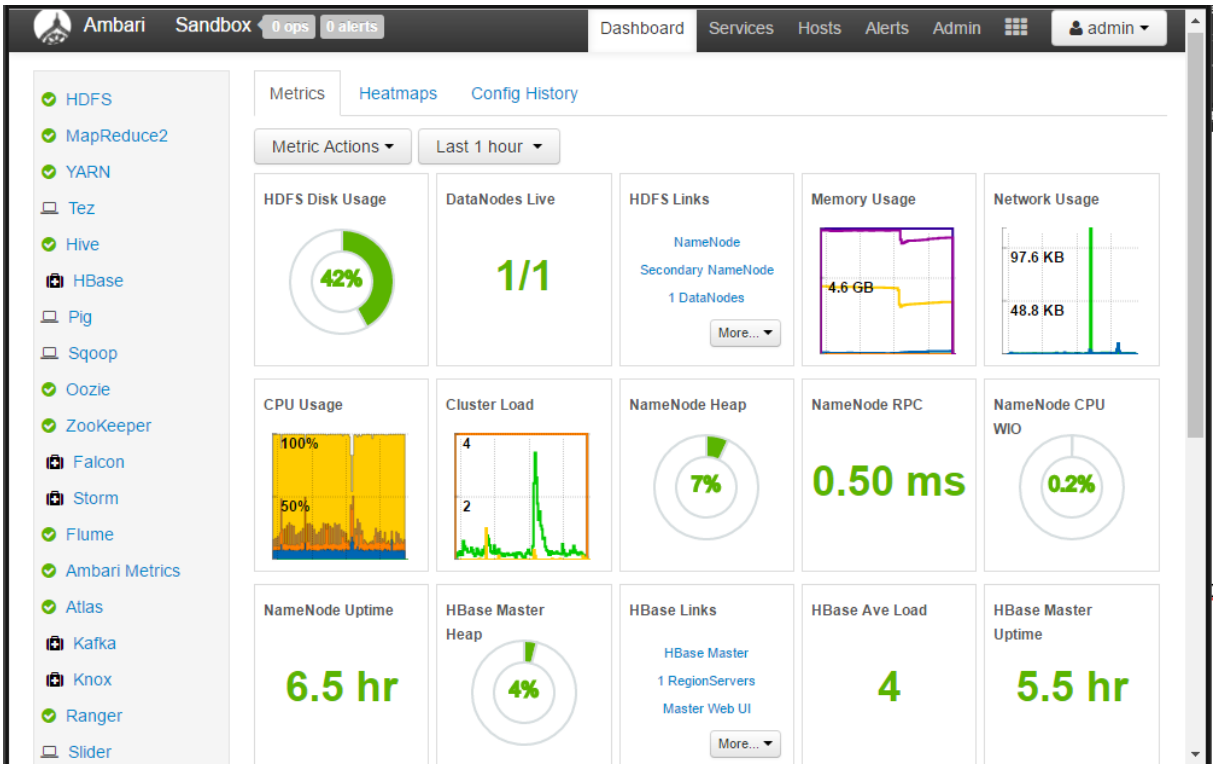
Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
Accumulo	TCP	147.96.25.33	50095		50095
AmbariShell	TCP	147.96.25.33	4200		4200
Atlas	TCP	147.96.25.33	21000		21000
Datanode	TCP	147.96.25.33	50075		50075
Falcon	TCP	147.96.25.33	15000		15000
HBaseMaster	TCP	147.96.25.33	16010		16010
HBaseRegion	TCP	147.96.25.33	16030		16030
HS2	TCP	147.96.25.33	10000		10000
HS2Http	TCP	147.96.25.33	10001		10001
HST	TCP	147.96.25.33	9000		9000
JobHistory	TCP	147.96.25.33	19888		19888
Knox	TCP	147.96.25.33	8443		8443
Nifi	TCP	147.96.25.33	9090		9090
NodeManager	TCP	147.96.25.33	8042		8042
Oozie	TCP	147.96.25.33	11000		11000
RM	TCP	147.96.25.33	8050		8050
RServer	TCP	147.96.25.33	8797		8787
Solr	TCP	147.96.25.33	8993		8993
SolrAdmin	TCP	147.96.25.33	8983		8983
Spark	TCP	147.96.25.33	4040		4040
SparkHistoryServer	TCP	147.96.25.33	18080		18080
StormUI	TCP	147.96.25.33	8744		8744
Tutorials	TCP	147.96.25.33	8888		8888
WebHBase	TCP	147.96.25.33	60080		60080
WebHcat	TCP	147.96.25.33	50111		50111

El siguiente paso será asignar una nueva contraseña al usuario administrador de Ambari, para ello ingresamos como usuario *root* a la máquina virtual, donde por primera vez también nos pedirá cambiar la contraseña por defecto de este usuario, y ejecutamos la orden *ambari-password-reset*.

```
[root@sandbox ~]# ambari-admin-password-reset
Please set the password for admin:
Please retype the password for admin:

The admin password has been set.
Restarting ambari-server to make the password change effective...

Using python  /usr/bin/python2
Restarting ambari-server
Using python  /usr/bin/python2
Stopping ambari-server
Ambari Server stopped
Using python  /usr/bin/python2
Starting ambari-server
Ambari Server running with administrator privileges.
Organizing resource files at /var/lib/ambari-server/resources...
Server PID at: /var/run/ambari-server/ambari-server.pid
Server out at: /var/log/ambari-server/ambari-server.out
Server log at: /var/log/ambari-server/ambari-server.log
Waiting for server start.....................
Ambari Server 'start' completed successfully.
[root@sandbox ~]#
```

Una vez realizados los cambios de contraseña podemos ingresar en el modo administrador del servicio Ambari en la que tenemos una interfaz amigable donde poder realizar las implementaciones y configuraciones de las demás aplicaciones a usar dentro del entorno Hadoop.





Apéndice: Archivo de configuración de Flume

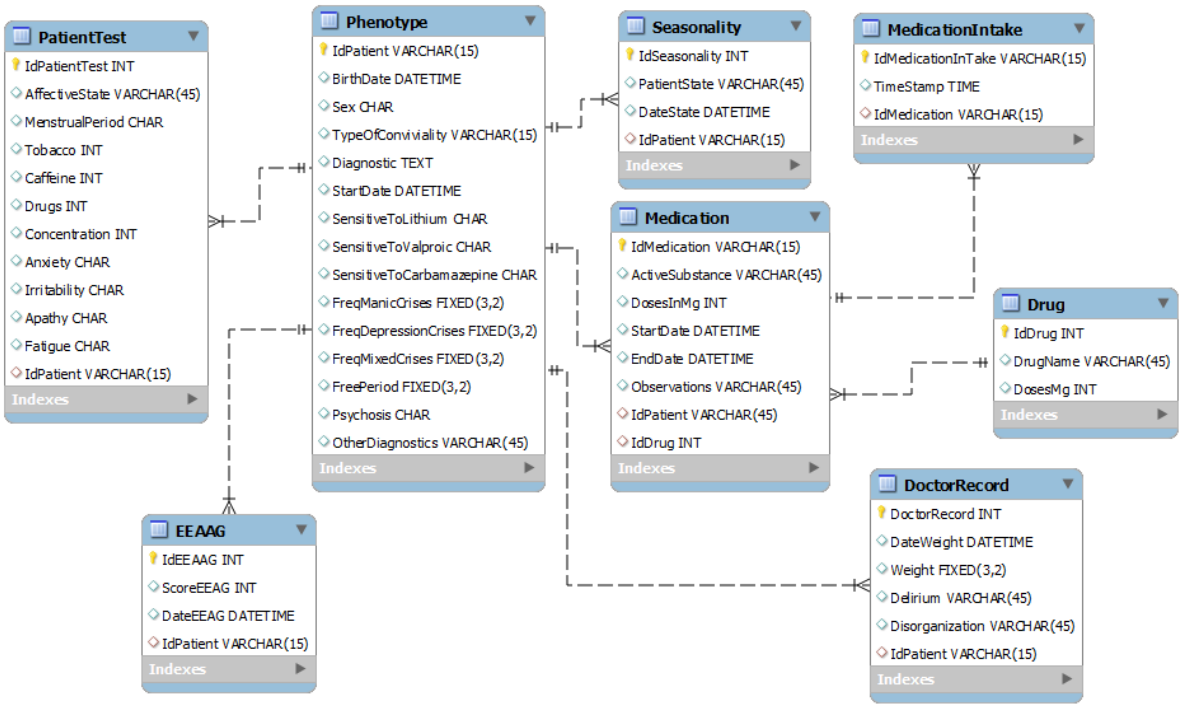
```
# Define a source, a channel, and a sink
agent.sources = src1
agent.channels = chan1
agent.sinks = sink1

# Set the source type to Spooling Directory and set the directory
# location to /media/sf_toHDFS/

agent.sources.src1.type = spooldir
agent.sources.src1.spoolDir = /media/sf_toHDFS/
agent.sources.src1.basenameHeader = true
agent.sources.src1.deserializer = org.apache.flume.sink.solr.morphline.BlobDeserializer$Builder

# Configure the channel as simple in-memory queue
agent.channels.chan1.type = memory
agent.channels.chan1.capacity = 2000

# Define the HDFS sink and set its path to your target HDFS directory
agent.sinks.sink1.type = hdfs
agent.sinks.sink1.hdfs.path = hdfs://sandbox.hortonworks.com:8020/userdata //*
agent.sinks.sink1.hdfs.fileType = DataStream

# Disable rollover functionallity as we want to keep the original files
agent.sinks.sink1.rollCount = 0
agent.sinks.sink1.rollInterval = 0
agent.sinks.sink1.rollSize = 0
agent.sinks.sink1.idleTimeout = 0

# Set the files to their original name
agent.sinks.sink1.hdfs.filePrefix = %{basename}

# Connect source and sink
agent.sources.src1.channels = chan1
agent.sinks.sink1.channel = chan1
```

Para conocer la ruta del puerto activo de hdfs el cual debe ir en *agent.sinks.sink1.hdfs.path* ejecutamos los siguientes comandos:

hdfs getconf -confKey fs.default.name

Apéndice: Modelo relacional de la base de datos PCB

El siguiente modelo relacional fue diseñado para capturar la información relevante de los pacientes diagnosticados con el trastorno de bipolaridad.



Apéndice: Instalación de R y R Studio en Hortonworks

- Agregar una regla con el puerto 8787 en el reenvío de puertos de VirtualBox
- Instalar R

  yum install R
- Añadir RHadoop que es una colección de paquetes que ayuda a aprovechar el entorno Hadoop en operaciones con grandes volúmenes de datos.
- Por último, se instala R Studio, al cual se podrá acceder desde el navegador en un puerto específico como se hacen con los servicios de Hortonworks.

  #Descargar R Studio

  wget -O /tmp/rstudio-server-0.98.1091-x86_64.rpm

  http://download2.rstudio.org/rstudio-server-0.98.1091-x86_64.rpm

  #Instalar R Studio suprimiendo la verificación de firmas del paquete

  sudo yum -y install --nogpgcheck /tmp/rstudio-server-0.98.1091-x86_64.rpm

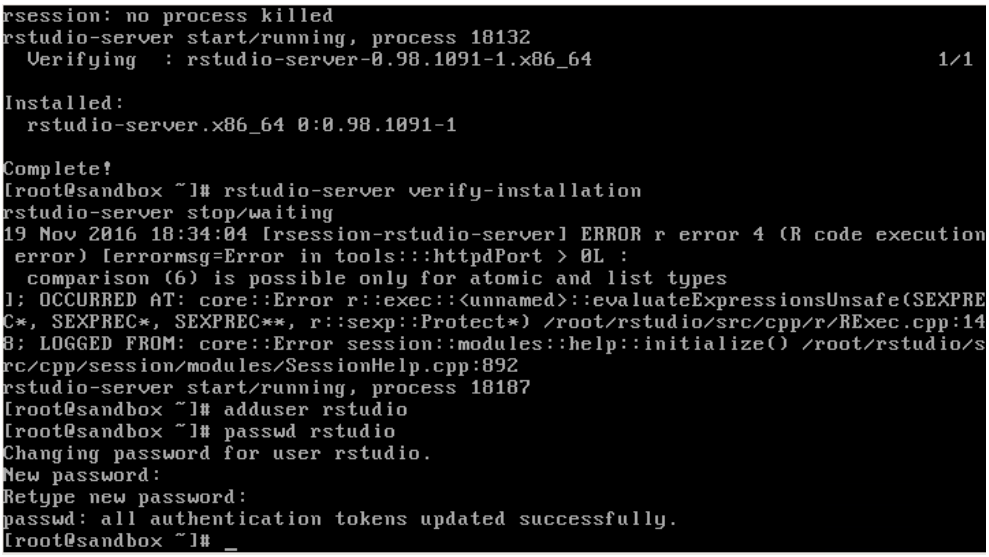
  # Verificación de la instalación

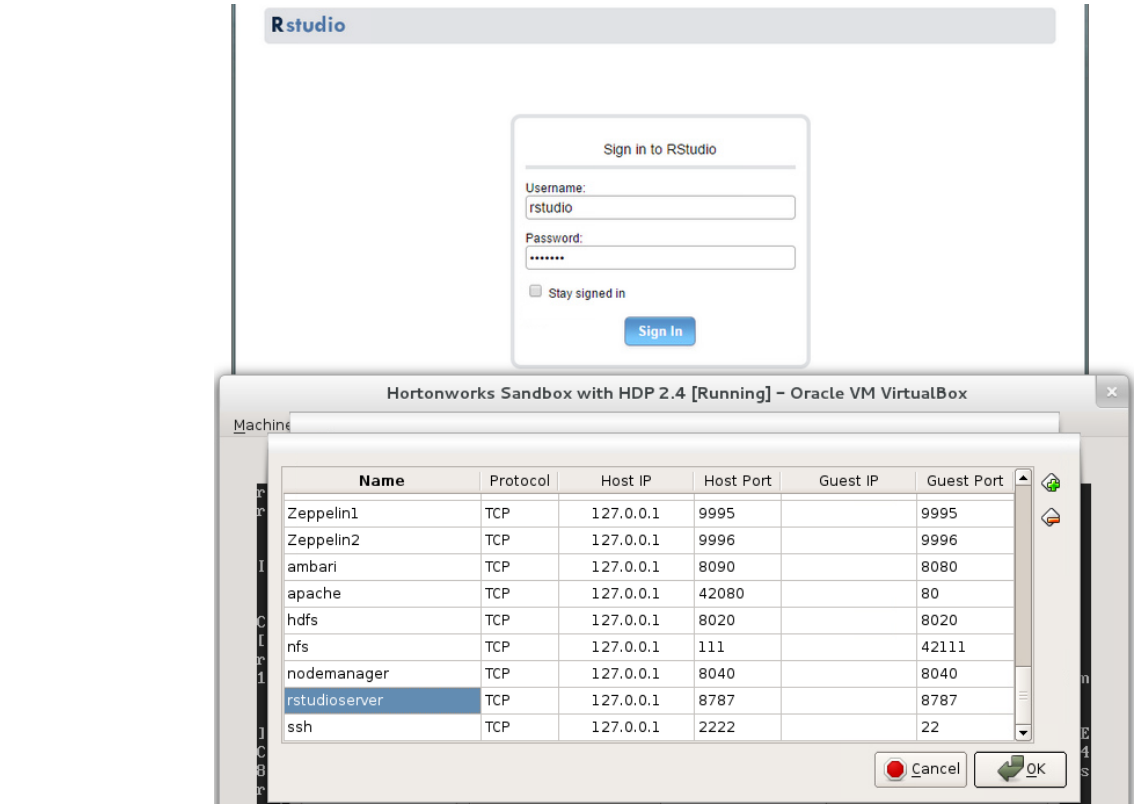
  sudo rstudio-server verify-installation

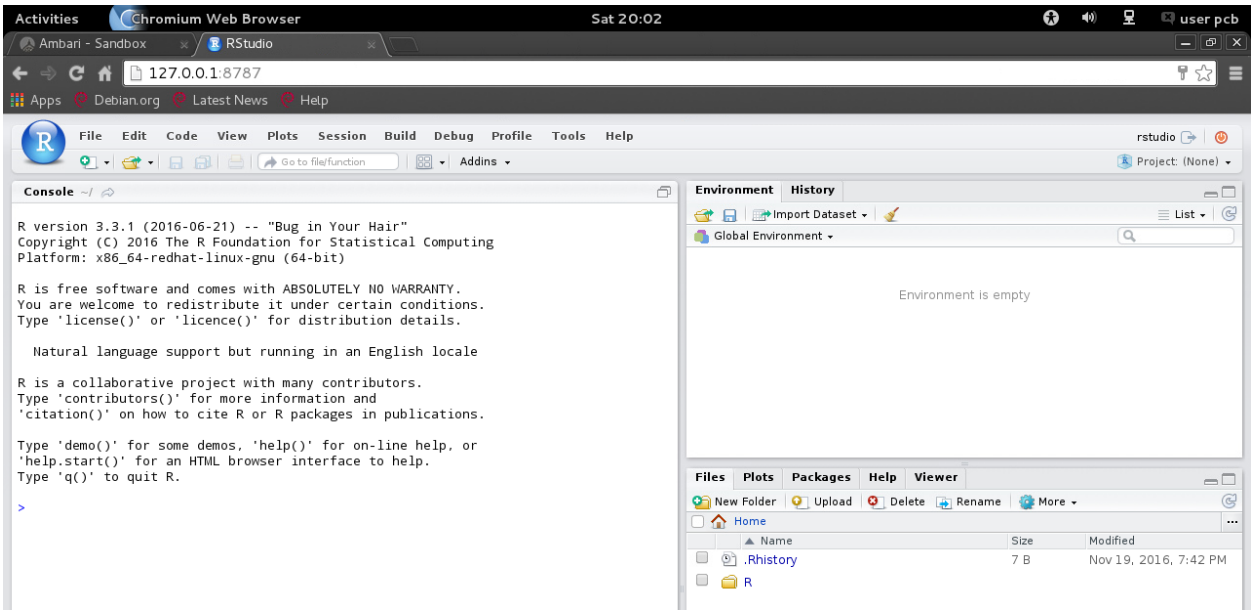
  # Agregar un usuario de acceso a R Studio

  sudo adduser rstudio

  sudo passwd rstudio

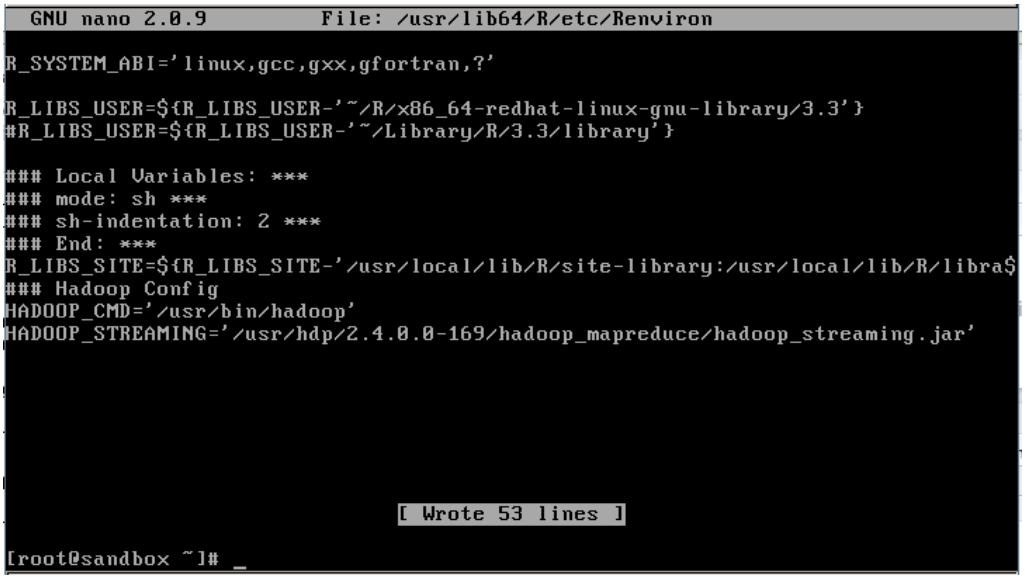




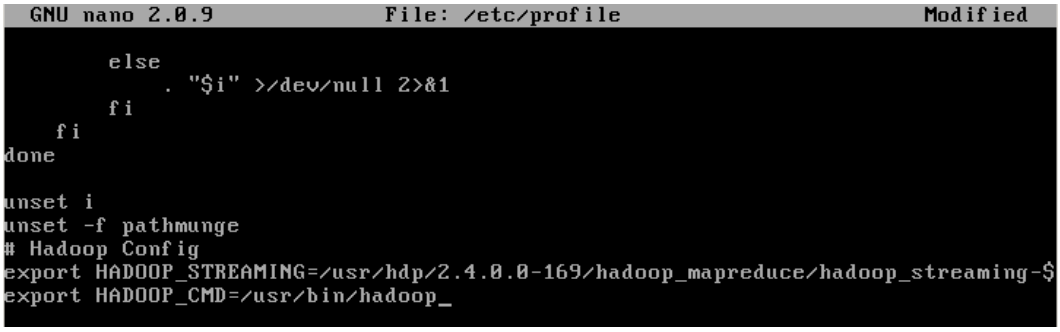


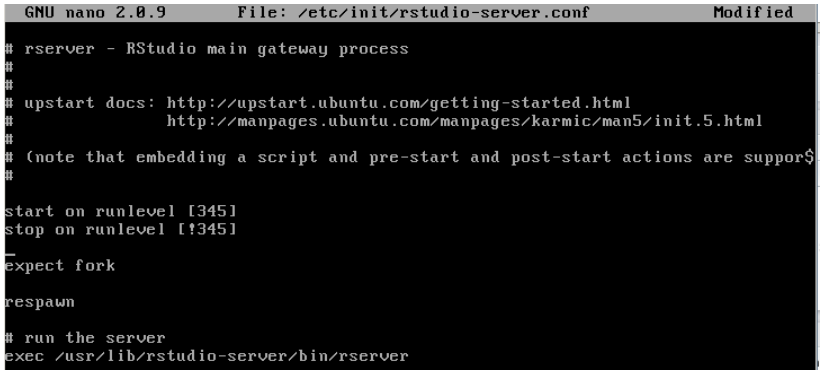
Buscamos el nombre y la versión de los ficheros Hadoop streaming para poder configurarlos en R.

```
[root@sandbox ~]# find / -name "hadoop-streaming*.jar"
/usr/hdp/2.4.0.0-169/hadoop-mapreduce/hadoop-streaming-2.7.1.2.4.0.0-169.jar
/usr/hdp/2.4.0.0-169/hadoop-mapreduce/hadoop-streaming.jar
/usr/hdp/2.4.0.0-169/oozie/share/lib/mapreduce-streaming/hadoop-streaming-2.7.1.
2.4.0.0-169.jar
[root@sandbox ~]#
```



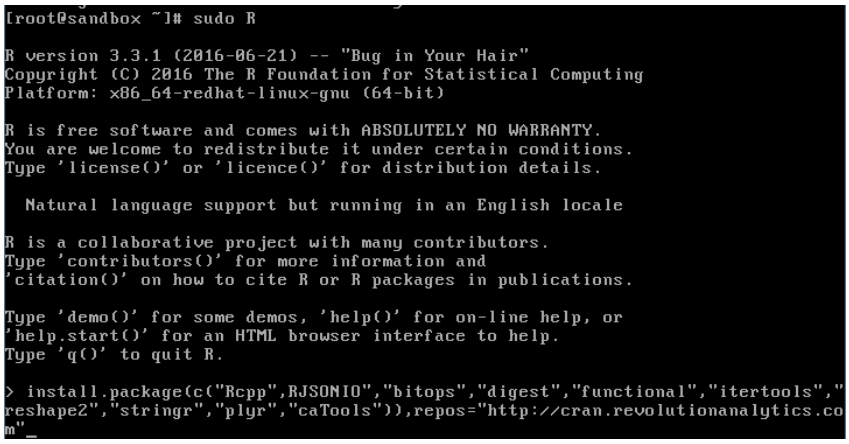

```
Sys.setenv("HADOOP_CMD"="/usr/hdp/2.3.2.0-2950/hadoop/bin/yarn")

Sys.setenv("HADOOP_STREAMING"="/usr/hdp/2.3.2.0-2950/hadoop-mapreduce/hadoop-
streaming-2.7.1.2.3.2.0-2950.jar")

Sys.setenv("HADOOP_HOME"="/usr/hdp/2.3.2.0-2950/hadoop")

Sys.setenv("HADOOP_PREFIX"="/usr/hdp/2.3.2.0-2950/hadoop")
```





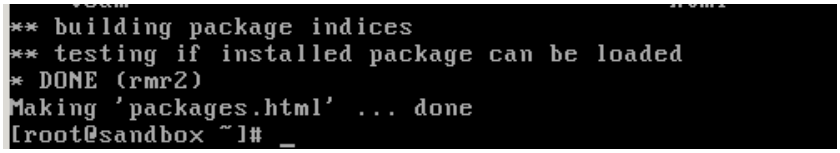
Instalar los paquetes necesarios en R

```
install.packages(c("Rcpp","RJSONIO","bitops","digest","functional","itertools","reshape2","stringr","plyr","caTools"),repos='http://cran.revolutionanalytics.com')
```

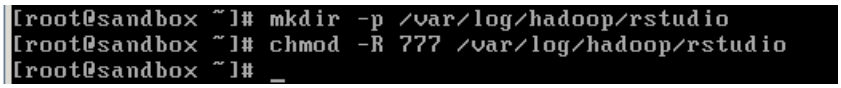


Instalación del paquete rmr2 en R, que nos permitirá lanzar MapReduce:

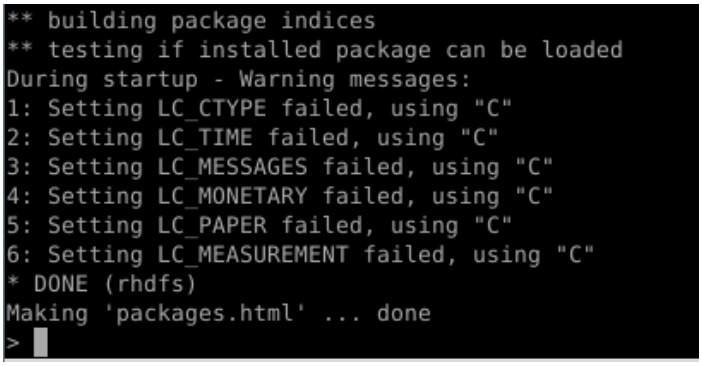
```
wget -O /tmp/rmr2_3.3.0.tar.gz https://github.com/RevolutionAnalytics/rmr2/raw/master/build/rmr2_3.3.0.tar.gz
sudo R CMD INSTALL /tmp/rmr2_3.3.0.tar.gz
```



Creación del directorio donde se tendrán los log de RStudio.



Instalación del paquete rhdfs



Carga de las librerías instaladas en R.

```
** testing if installed package can be loaded
During startup - Warning messages:
1: Setting LC_CTYPE failed, using "C"
2: Setting LC_TIME failed, using "C"
3: Setting LC_MESSAGES failed, using "C"
4: Setting LC_MONETARY failed, using "C"
5: Setting LC_PAPER failed, using "C"
6: Setting LC_MEASUREMENT failed, using "C"
* DONE (rhdfs)
Making 'packages.html' ... done
> library(rJava)
> library(rmr2)
Please review your hadoop settings. See help(hadoop.settings)
> library(rhdfs)

HADOOP_CMD=/usr/hdp/2.4.0.0-169/hadoop/bin/yarn

Be sure to run hdfs.init()
> hdfs.init()
16/11/21 23:17:53 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
16/11/21 23:17:54 WARN shortcircuit.DomainSocketFactory: The short-circuit local
 reads feature cannot be used because libhadoop cannot be loaded.
```

Prueba de MapReduce en R:

SampleMR.R

Source on Save | Run | Source

```
1  library(rJava)
2  library(rmr2)
3  library(rhdfs)
4  hdfs.init()
5  input = to.dfs(1:10)
6  driver = mapreduce(input = input, map = function(k, v) cbind(v, 2*v))
7  tushar <- from.dfs(driver)
8  tushar$val
```

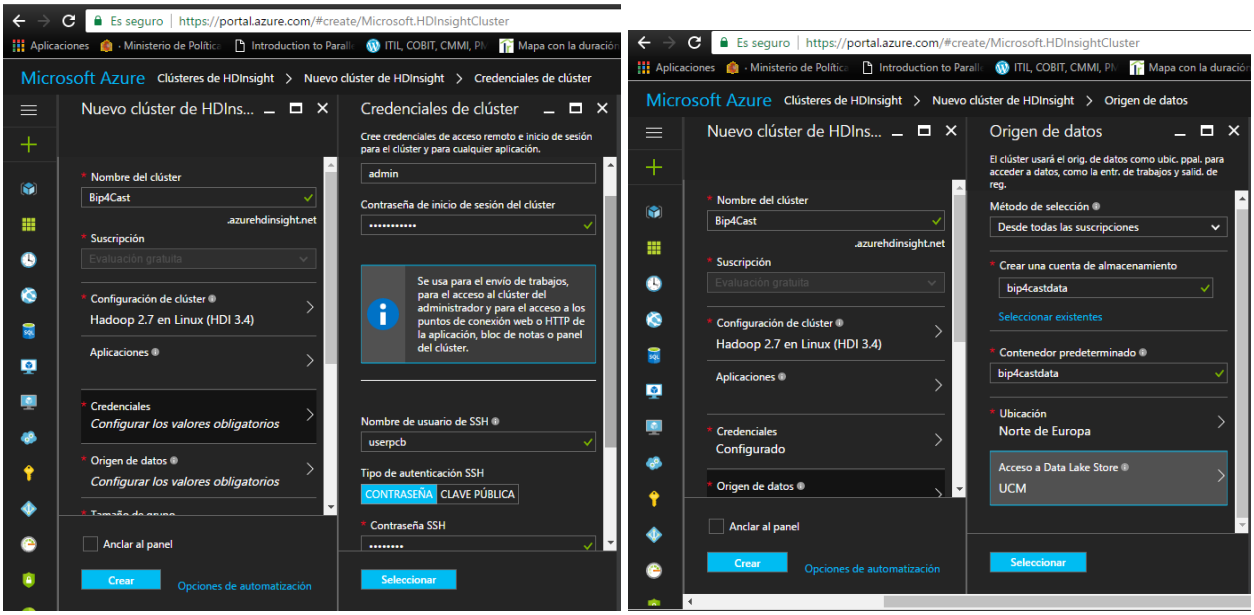
8:11 (Top Level) R Script

Console ~/

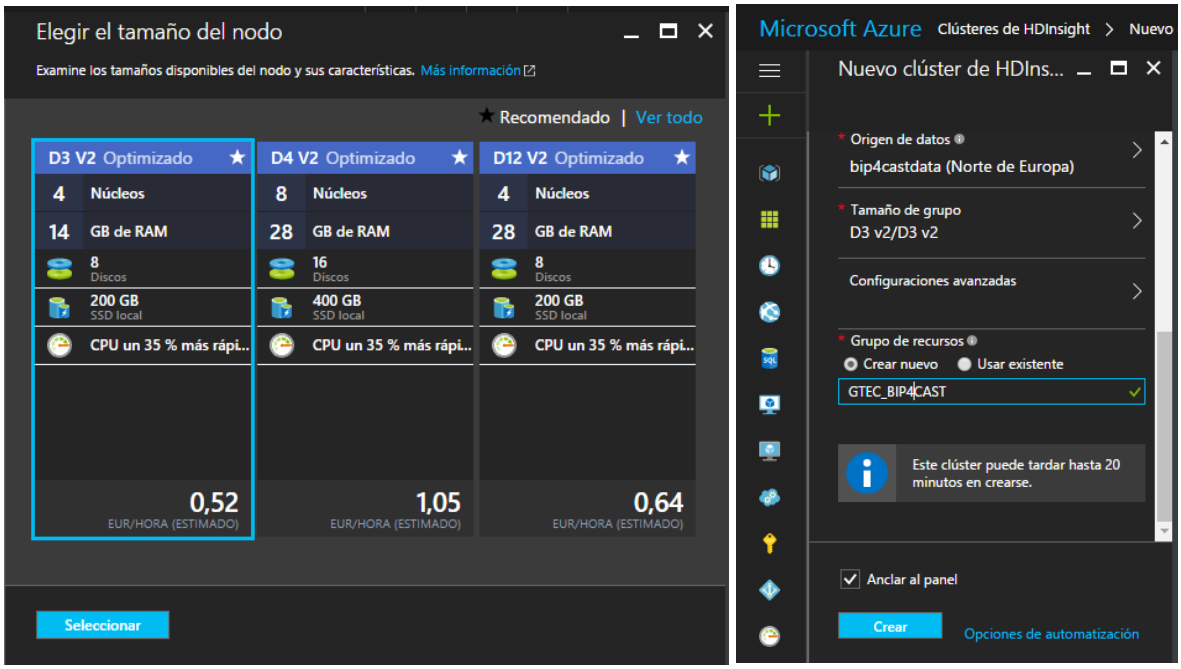
```
                Total committed heap usage (bytes)=306184192
        File Input Format Counters
                Bytes Read=793
        File Output Format Counters
                Bytes Written=919
16/11/21 23:33:21 INFO streaming.StreamJob: Output directory: /tmp/file54773b59ddaa
> tushar$val
       v
 [1,]  1  2
 [2,]  2  4
 [3,]  3  6
 [4,]  4  8
 [5,]  5 10
 [6,]  6 12
 [7,]  7 14
 [8,]  8 16
 [9,]  9 18
[10,] 10 20
```

Apéndice: Implementación de HDInsight - Azure

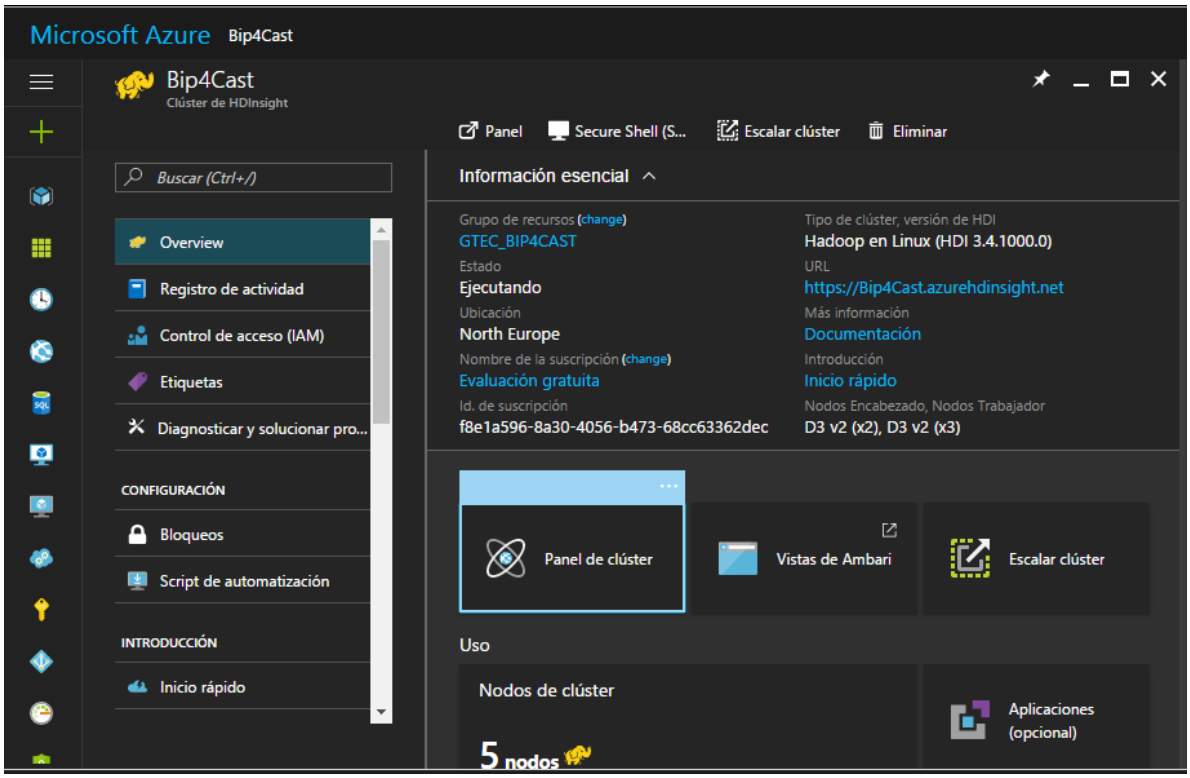
Configuración del clúster:



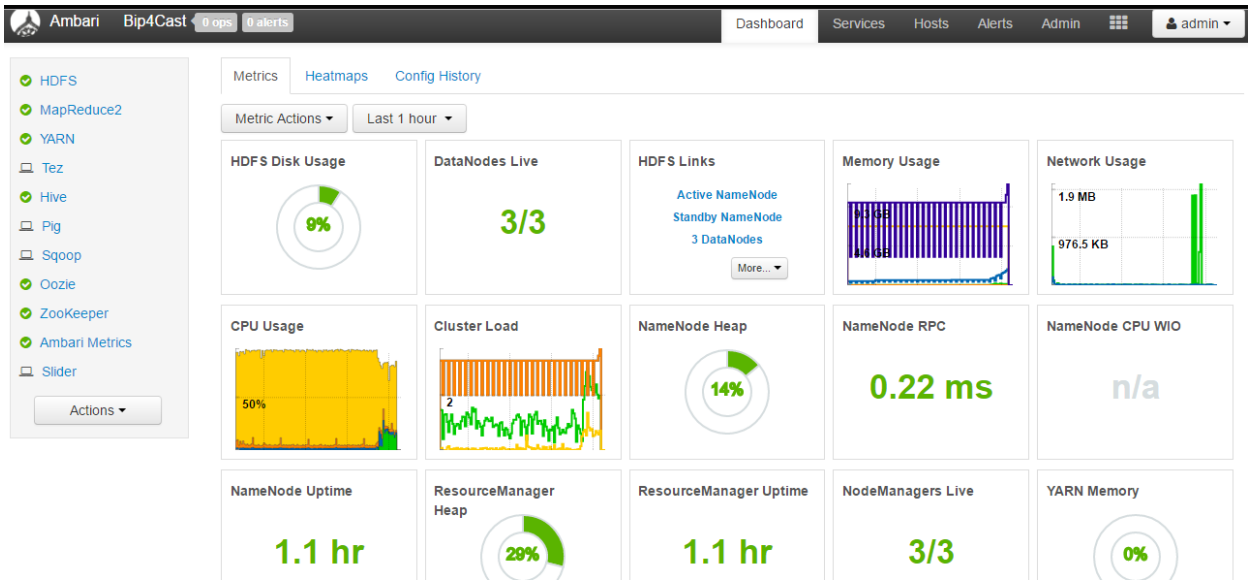
Elección del tipo de nodo a contratar:



Panel principal del clúster implementado:



Panel de gestión de Ambari:

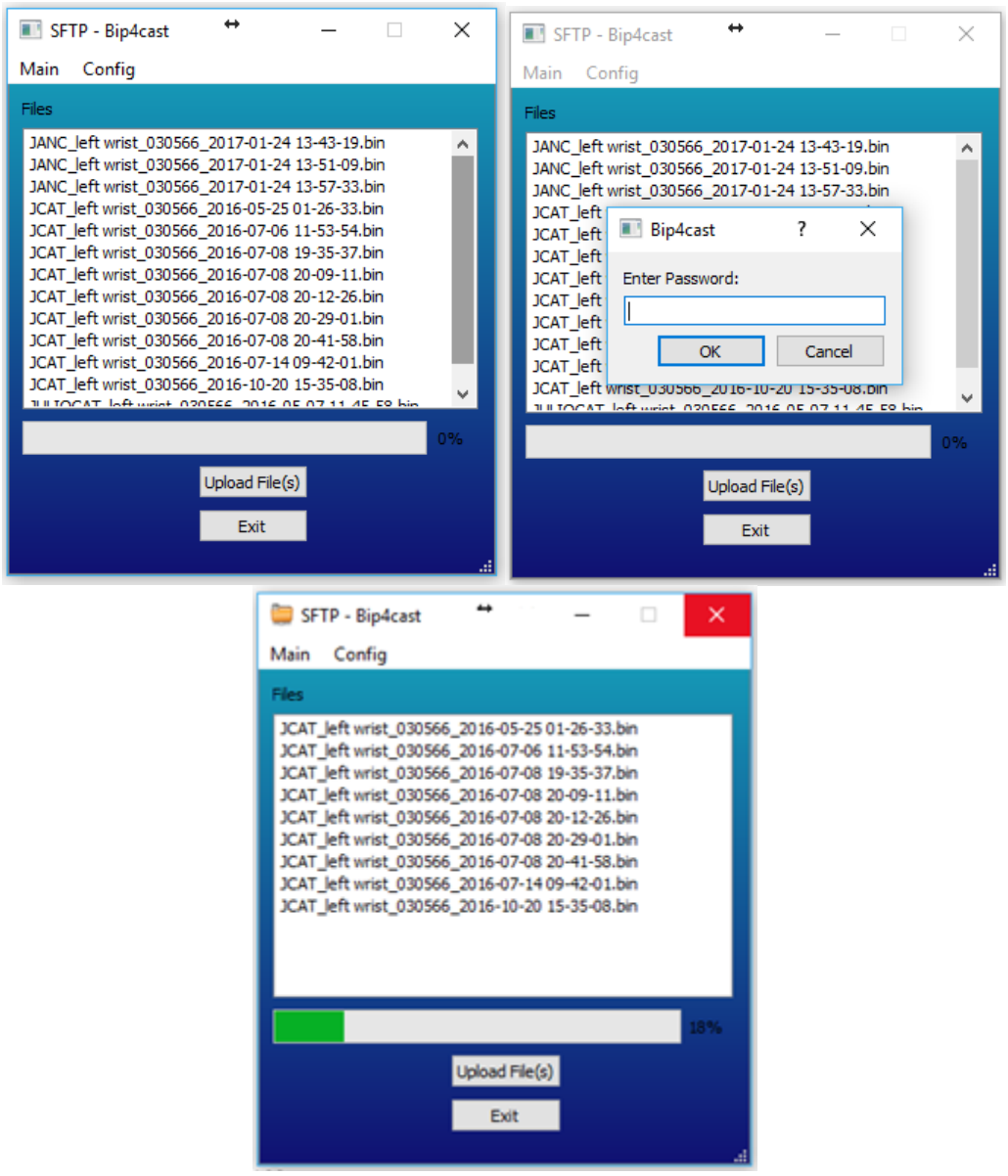


Apéndice: Código para el envío de archivos por medio SFTP desde GENEActiv PC Software.

```
using Tamir.SharpSsh; //Librería SFTP para C#
//Método subir archivos
private static void FileUploadUsingSftp(string SFTPAddress, string SFTPUserName, string SFTPPassword, string
SFTPFilePath, string FileName)
   {
      Sftp sftp = null;
      try
      {
         sftp = new Sftp(SFTPAddress, SFTPUserName, SFTPPassword);
         // Connect Sftp
         sftp.Connect();
         AutoClosingMessageBox.Show("Connected! Sending Data", "Upload Data", 2000);
         // upload file
         string strippedFileName = Folders.GetDataFolder()+"\\"+FileName;
         sftp.Put(strippedFileName, "//userdata//" + FileName);
         AutoClosingMessageBox.Show("Uploaded! Now configure the device...", "Upload Data", 3000);
         sftp.Close(); // Close the Sftp connection
      }
      catch (Exception ex)
      {
         MessageBox.Show(ex.ToString());
      }
      finally
      {
         if (sftp != null)
         { sftp.Close(); }
      }
   }
    //Messagebox with timer
   public class AutoClosingMessageBox
   {
      System.Threading.Timer _timeoutTimer;
      string _caption;
      AutoClosingMessageBox(string text, string caption, int timeout)
      {
         _caption = caption;
         _timeoutTimer = new System.Threading.Timer(OnTimerElapsed,
            null, timeout, System.Threading.Timeout.Infinite);
         MessageBox.Show(text, caption);
      }
      public static void Show(string text, string caption, int timeout)
      {  new AutoClosingMessageBox(text, caption, timeout);
      }
      void OnTimerElapsed(object state)
      {
         IntPtr mbWnd = FindWindow(null, _caption);
         if (mbWnd != IntPtr.Zero)
            SendMessage(mbWnd, WM_CLOSE, IntPtr.Zero, IntPtr.Zero);
         _timeoutTimer.Dispose();
      }
      const int WM_CLOSE = 0x0010;
      [System.Runtime.InteropServices.DllImport("user32.dll", SetLastError = true)]
      static extern IntPtr FindWindow(string lpClassName, string lpWindowName);
      [System.Runtime.InteropServices.DllImport("user32.dll", CharSet = System.Runtime.InteropServices.CharSet.Auto)]
      static extern IntPtr SendMessage(IntPtr hWnd, UInt32 Msg, IntPtr wParam, IntPtr lParam); }   }
```

Apéndice: Aplicación cliente SFTP con Python y PyQt

El código fuente se encuentra en GitHub: https://github.com/juliocat/Bip4CastBigData



Apéndice: Algoritmo MapReduce para el análisis de la actividad de los pacientes

```
from mrjob.job import MRJob
from math import sqrt

class MRWordCount(MRJob):
  # Fase MAP
  def mapper(self, key, line):
    items = line.split(",")
    values=[items[x] for x in range(1, 11)]
    #Send k=day-hour and v=items to calc
    yield items[0][:14], [float(i) for i in values]

  # Fase REDUCE x day-hour
  def reducer(self, key, values):
    x=list(values)
    totalit=len(x)
    ###Activity Base Lines###
    sedentary_light=410
    light_moderate=576
    moderate_vigorous=1924
    ######
    #Variables to calc
    current_mov=0.0
    crt_svm=0.0
    hour_avg=0.0
    min_hravg=0.0
    nowear=False
    activity=""
    lstcrt_svm=[]
    lshour_avg=[]
    lsmin_hravg=[]
    lsnowear=[]
    lsactivity=[]
    for i in range(0,totalit):
      #Current Movement
      if i==0:
        current_mov=0
      current_mov=sqrt(abs(x[i-1][0]-x[i][0]+x[i-1][1]-x[i][1]+x[i-1][2]-x[i][2]))
      # Current movement * SVM
      crt_svm=current_mov*x[i][6]
      lstcrt_svm.append(crt_svm)
    # 1 hour average
```

```
for i in range(0,totalit):
  if i<=30:
    aux_hravg=[lstcrt_svm[i] for i in range(0,totalit-31)]
  else:
    aux_hravg=[lstcrt_svm[i] for i in range(i,len(lstcrt_svm))]
  hour_avg=sum(aux_hravg)/len(aux_hravg)
  lshour_avg.append(hour_avg)
# Min 1 Hour average
for i in range(0,totalit):
  if i<=30:
    auxmin_hravg=[lshour_avg[i] for i in range(0,totalit-31)]
  else:
    auxmin_hravg=[lshour_avg[i] for i in range(i,len(lshour_avg))]
  min_hravg=min(auxmin_hravg)
  lsmin_hravg.append(min_hravg)
  # No wear
  nowear=min_hravg<1
  lsnowear.append(nowear)
#Activity
for i in range(0,totalit):
  if lsnowear[i]==True:
    lsactivity.append("nowear")
  else:
    if x[i][6]>moderate_vigorous:
      lsactivity.append("vigorous")
    else:
      if x[i][6]>light_moderate:
        lsactivity.append("moderate")
      else:
        if x[i][6]>sedentary_light:
          lsactivity.append("light")
        else:
          lsactivity.append("sedentary")

yield key, lsactivity
```