DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS E INFORMÁTICA

TEMA: ANÁLISIS DE PATRONES DE COMPORTAMIENTO APLICANDO TÉCNICAS DE BIG DATA USANDO DATASTREAM EN INVERNADEROS

AUTOR: GORDON CERNA, DALIA CAROLINA

DIRECTOR: Ing. DÍAZ ZÚÑIGA, MAGI PAÚL Msc, MBA.

SANGOLQUÍ
2018

CERTIFICADO

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

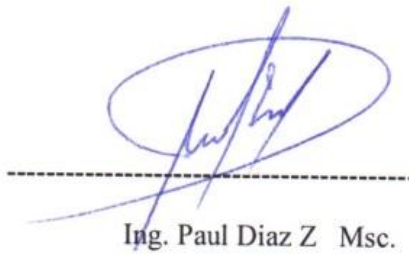
CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

CERTIFICACIÓN

Certifico que el trabajo de titulación, "**ANÁLISIS DE PATRONES DE COMPORTAMIENTO APLICANDO TÉCNICAS DE BIG DATA USANDO DATASTREAM EN INVERNADEROS**" fue realizado por la señorita **DALIA CAROLINA GORDON CERNA** el mismo que sido revisado en su totalidad y analizado por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 07 de agosto de 2018

Firma:



Ing. Paul Diaz Z Msc.

1707249072

AUTORÍA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

AUTORÍA DE RESPONSABILIDAD

Yo, **GORDON CERNA, DALIA CAROLINA**, declaro que el contenido, ideas y criterios del trabajo de titulación: "**ANÁLISIS DE PATRONES DE COMPORTAMIENTO APLICANDO TÉCNICAS DE BIG DATA USANDO DATASTREAM EN INVERNADEROS**" es de mi autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Consecuentemente el contenido de la investigación mencionada es veraz.

Sangolquí, 06 de agosto de 2018

Firma:

DALIA CAROLINA GORDON CERNA

C.C. 1803217437

AUTORIZACIÓN

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

AUTORIZACIÓN

Yo, **GORDON CERNA, DALIA CAROLINA** autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación "**ANÁLISIS DE PATRONES DE COMPORTAMIENTO APLICANDO TÉCNICAS DE BIG DATA USANDO DATASTREAM EN INVERNADEROS**" en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 06 de agosto de 2018

Firma:

DALIA CAROLINA GORDON CERNA

C.C. 1803217437

DEDICATORIA

Le dedico este proyecto de investigación a mis padres que desde siempre me han enseñado el valor del trabajo duro y la dedicación con el ejemplo, a mis hermanas que siempre han estado conmigo apoyándome y ayudándome en todo lo que necesito.

A todos quienes han estado a mi lado en este trayecto de la universidad ya que, en algún momento y a pesar de todas las dificultades que teníamos, nos dimos la mano para superarnos a nosotros mismos y sacar adelante nuestras metas.

AGRADECIMIENTO

Agradezco a Dios por las bendiciones brindadas y por permitirme seguir cumpliendo las metas que me he propuesto.

Agradezco a mis papis y a mis hermanas, porque me han dado todo el apoyo que he necesitado en todos los sentidos y sé que cuento con ellos para lo que sea.

A mi novio que me ha estado empujando hacia adelante a cada momento hasta alcanzar lo que me propuesto y a no desfallecer en el camino.

Al ingeniero Paúl Díaz, porque gracias a su guía incondicional me ha permitido culminar la carrera con éxito y cumplir más de una meta.

A todos mis familiares, amigos y compañeros que me dieron fuerzas, tiempo y palabras de aliento para continuar.

INDICE

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN

En la actualidad, en casi cualquier campo de trabajo, se necesita procesar datos de manera más efectiva e instantánea. Uno de los campos que en Ecuador es imperativo se mejore, es el control en invernaderos para el correcto tratamiento de cultivos, sobre todo de exportación. En el campus IASA I de la Universidad de las Fuerzas Armadas ESPE, se lleva a cabo un proyecto para precisamente mejorar el control de su invernadero de rosas, dentro de este proyecto se han buscado las mejores herramientas en el campo de flujos de datos continuos con Big data como parte de la respuesta. Se identificaron dos herramientas idóneas para el tratamiento de flujos de datos provenientes del invernadero que son Spark Streaming y Apache Flink, se realizó una comparativa basándose en la norma ISO 25000 y se obtuvo como resultado que la herramienta Spark Streaming fue la idónea para el procesamiento de estos datos. Los datos para la solución que brinda Spark streaming, se encuentran almacenados en archivos y fueron procesados de manera que se discriminaba datos correctos o incorrectos de acuerdo a la temperatura y hora que fue trasmitida desde uno de los sensores dispuestos en el invernadero. Como resultado se encuentra que efectivamente la herramienta fue la indicada para los flujos de datos, y se anima a realizar más verificaciones y a interactuar con Spark streaming y otros de los sensores de las rosas, siempre con la primicia de mejorar el control del mismo.

PALABRAS CLAVES

- **BIG DATA**
- **SPARK STREAMING**
- **APACHE FLINK**
- **INVERNADEROS**
- **FLUJOS DE DATOS CONTINUOS**

ABSTRACT

Nowadays, almost in any field of work, it is needed to process data effectively and instantaneously. One of the imperative fields of improvement in Ecuador is the control of greenhouses for the correct treatment of crops, especially for export. On the IASA I campus of the University of the Armed Forces ESPE, a project is being carried out to precisely improve the control of its roses greenhouse, within this project the best tools in the field of data streaming with Big Data, have been searched as part of the response. Two suitable tools were identified for the treatment of datastream from the greenhouse, Spark Streaming and Apache Flink, a comparison was made based on the ISO 25000 standard and the result was that the Spark Streaming tool was the ideal tool for processing these dates. The data for the solution provided by spark streaming, are stored in files and were process in a way that we discriminate them like correct or incorrect according to the temperature and time that was transmitted from one of the sensors arranged in the greenhouse. As a result, we found that the tool was indeed the one indicated for this datastream, and we encourage the community to carry out more verifications and to interact with Spark streaming and other sensors of the roses, always with the scoop of improving the control.

KEY WORDS

- **BIG DATA**
- **SPARK STREAMING**
- **APACHE FLINK**
- **GREENHOUSES**
- **DATA STREAMING**

CAPITULO I
INTRODUCCIÓN

1.1 ANTECEDENTES

Con el apoyo del Instituto Agropecuario de la ESPE - IASA I y su centro de investigación de cultivos de rosas, dirigido por la Dra. Elizabeth Urbano experta en este tipo de ámbito; se está llevando a efecto varios proyectos en conjunto con la Carrera de Ingeniería de Sistemas ESPE, encabezada por el Ing. Paúl Díaz, cuyo principal objetivo es buscar alternativas de mejora a la producción mediante técnicas de streaming, minería e implementación de una red de Wireless Sensor Network que emitan datos de los cultivos indicados y así poder predecir ciertas anomalías y mejoras.

Dicho proyecto tiene varias etapas y objetivos claros, dentro de los cuales se contempla el obtener una solución tecnológica directamente para los productores ecuatorianos de rosas en la toma acertada de decisiones de entrega oportuna del producto dependiendo la temporada.

Además, en países más desarrollados en cultivos de invernaderos como España, existe una gama mucho más completa sobre el control en un invernadero. Esto es observable en la revista Horticon.com que se especializa en tecnología de horticultura con el artículo “Sensores para el control climático en invernadero” escrito por investigadores de la Universidad Politécnica de Madrid; en donde se hace un compendio de todos los sensores con los que ellos han trabajado y mencionan que “cada medida debe ser registrada y manejada (…) a través de un controlador y ordenador, de forma que el propio usuario pueda verificar y modificar las consignas según las condiciones climáticas requeridas en cada momento” (Perdigones, Peralta, Nolasco, Muños, & Pascual, 2004)

Correspondiendo con lo anterior, el estudio se ha extendido en otros proyectos investigativos, que por lo general planean contribuir con un modelo predictivo aplicando técnicas de minería de datos y/o técnicas de Big data, donde los datos se obtendrán previamente de sensores inalámbricos puestos a disposición en un entorno cambiante, en este caso un invernadero. En

estos proyectos, junto con expertos en el área de estudio, se realizarán pruebas de campo y posteriormente se clasificará y almacenará la información obtenida para ponerla a disposición.

1.2 PROBLEMÁTICA

Ecuador es un país eminentemente agrícola y por tanto cualquier esfuerzo en mejorar sus procesos productivos no es un desperdicio. Si de esto se parte que existe una cultura empírica para la producción y tratamiento en invernaderos, se podría crear una base de datos de este conocimiento y condiciones, crear bases de datos reales que ayuden al mejoramiento de la producción en los invernaderos y al análisis medioambiental.

Una de las problemáticas de los agricultores ecuatorianos, es el enfrentarse a un escenario donde su experiencia no haya sido documentada o donde no existe el conocimiento suficiente sobre las condiciones en la producción dentro de invernaderos.

Según un reporte dado en el año 2003 por diario La hora, a pesar de la evidente expansión (de la exportación de flores) que se vivió en esa época, en muchos casos, ésta no fue de la mano con el desarrollo tecnológico, de allí se evidencia competencia desleal entre los propios floricultores ecuatorianos (La hora, 2003). Y en el año 2016, la misma fuente denota que la competencia sigue surgiendo, y ya no es únicamente competencia interna del país, si no competencia a nivel mundial o falta de interés debido a la crisis mundial. (La hora, 2016)

Para complementar, diario El Telégrafo en su reportaje de noviembre de 2017 titulado *El sector florícola no se recupera desde 2014* dice claramente "… en 2014 la industria florícola sufrió uno de sus peores momentos con la apreciación del dólar, la pérdida de competitividad del Ecuador y la caída del mercado ruso, uno de sus mayores compradores" (El Telégrafo, 2017).

Claramente se percibe que Ecuador debe ser más competitivo en la industria florícola, tanto a nivel interno como a nivel internacional. Dentro del plano tecnológico, los escenarios mencionados anteriormente dan pie a pensar que la información se trata de manera empírica casi en su totalidad, dando como resultado un retraso en el manejo y control de diferentes plantaciones como los invernaderos.

En vista de todo lo anteriormente mencionado, el presente trabajo de investigación pretende utilizar las técnicas más idóneas de Big data adaptables al comportamiento de la información proveniente de sensores inalámbricos o Wireless Sensor Network (WSN), como una práctica que podría solventar estas problemáticas, y con el avance del proyecto lograr, diferentes etapas, la automatización y control total sobre el invernadero.

1.3 JUSTIFICACIÓN

La idea de recolección de información y alimentación de una base de datos íntegra, podría no solo aumentar la calidad de la producción, sino que le daría al productor (agricultor) la facilidad de tomar decisiones confiables y a tiempo.

Supongamos entonces que se tienen varios sensores dentro de un invernadero que receptan datos específicos. Puede ser que, en el invernadero, los sensores se programen para receptar información cada cierto tiempo y que dicha información llegue en grandes cantidades permanentemente (flujos de datos). Para el correcto tratamiento de los diferentes flujos de datos, existen técnicas diferentes y es importante primero hacer pruebas en diferentes escenarios para localizar la mejor de éstas. Con esto se podría pensar en soluciones con Big data que ayuden a mejorar la experiencia de los productores de invernaderos y que con el tiempo esta práctica desemboque en datos verídicos

No se puede recolectar la información necesaria sin desperdiciar ninguno de los datos anteriormente desde la aplicación de técnicas de Big data especializadas en flujos de datos (datastream). Éstos harían fácil el procesar de manera fiable e instantánea los datos que llegan desde los invernaderos, y generar la información correcta en cualquier momento.

Aún mejor, la aplicación de estas técnicas de Big data ayudarían a almacenar información de condiciones ambientales reales que afecten directamente a los invernaderos y permitirían retroalimentar estos mismos datos.

1.4 OBJETIVOS

Objetivo General

Analizar patrones de comportamiento aplicando técnicas de Big data usando datastream en invernaderos.

Objetivos Específicos

- Realizar el análisis de dos técnicas para Big data.
- Aplicar la técnica de almacenamiento escalable de Big data y su indexación para búsquedas efectivas en datastream.
- Implementar una solución en Big data integrando la técnica analizada en ella.

1.5 ALCANCE

La propuesta parte desde que se han obtenido los datos procedentes del invernadero. Estos datos originan los flujos de datos que se requiere para poner a prueba la herramienta de Big data, puesto que existirán alrededor de 15000 datos por día, con un intervalo de toma de datos de cada 10 minutos.

Una vez obtenidos dichos flujos, se procede a escoger dos técnicas de Big data, correspondientemente se hablará de T1 (técnica 1) y T2 (técnica 2). Lo que permitirán estas técnicas es aplicar una serie de filtros (data cleaning) para escoger los datos más relevantes, convertirlos en información y, apoyados en el conocimiento del experto, transformarlos en sabiduría que admita tomar decisiones en tiempo y a tiempo.

Los datos provenientes de los WSN son: temperatura, humedad, brillo, índice de rayos UV y calidad de aire; puesto que son datos que se consideran comunes para cualquier tipo de invernaderos (sean éstos, por ejemplo, invernaderos de flores, frutas, etc.) e influyentes en la producción de cultivos sin importar su naturaleza.

CAPITULO II
ESTADO DE CUESTIÓN

Para desarrollar el presente trabajo de investigación se revisarán algunas herramientas de Big data y sus respectivas técnicas para el tratamiento de flujos de datos (datastream), es decir que se debe realizar un análisis de lo que puede ofrecer cada herramienta y cuál de ellas es la idónea. Para tener una idea clara de lo que se busca, será importante primero revisar algunos datos, conceptos, definiciones, etc. de lo que conlleva analizar técnicas de Big data.

2.1 BIG DATA

Big data puede ser un concepto (o varios conceptos) que provoca el interés de diversas ramas tanto en "ciencias computacionales, físicas, economía, matemática, ciencias políticas, bioinformática, sociología, y otras escuelas que claman por acceso a cantidades masivas de información" (Boyd & Crawford, 2012).

Por tanto, dentro de los muchos conceptos que pudiera darse a Big data podría destacar los brindados por Sam Madden en su investigación "From Databases to Big data", donde define que pueden ser "datos muy grandes, muy rápidos o muy difíciles para herramientas de procesamiento existentes" (Madden, 2012); por tanto ya se plantean nuevas formas de observar los datos y la necesidad de herramientas que soporten y permitan manejar de forma óptima las necesidades de Big data.

Para el año en que Madden escribió sobre Big data y la necesidad de más eficiencia, ya existían herramientas que manejaban éstas, pero de manera muy limitada. Entonces, se podría evidenciar el avance que ha existido, pero más aún, las necesidades crecientes en el manejo de los datos y su escalabilidad.

2.1.1 Las tres Vs de Big data

Otra consideración importante que se debe hacer con respecto a Big data es la generalización en las tres Vs. Las definiciones de las 3 Vs vendrían a remontar de cierto modo lo que Madden mencionó en sus conceptos; es decir que no se trataría únicamente de grandes volúmenes de datos, sino también de variedad y velocidad.

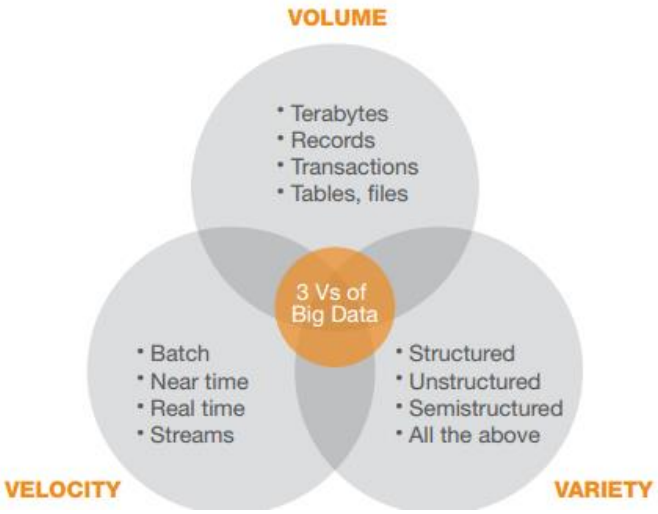


Figura 1. Las 3 Vs de Big data

Fuente: (Russom, 2011)

De lo observado en la Figura 1, se puede verificar que en la punta está el *Volumen* y que es fácil llegar a la conclusión que, de cualquier manera, la cantidad de datos es el "atributo primario de Big data" (Russom, 2011).

Otra definición importante es la *Variedad* de los datos, puesto que proceden de diferentes fuentes. Entre las principales se encuentran las que proceden de la Web, principalmente logs y medios sociales (redes sociales) (Russom, 2011).

Por último, se encuentra la *Velocidad*, que es referente a la "frecuencia de generación de los datos o la frecuencia en que se entregan" estos mismos datos (Russom, 2011). En este punto entra la discusión de flujos de datos que proceden de diferentes fuentes y necesitan ser analizadas en tiempo real, por ejemplo, lo que se ha hecho por años con las redes sociales donde se analizan las tendencias de los usuarios y se requiere de respuestas rápidas en los comportamientos.

A pesar de lo anterior, que es lo que convencionalmente define a Big data (es decir las tres Vs); existen autores que han agregado otras definiciones importantes como Mohammad Wazid, quien menciona brevemente entre las propiedades de Big data “volumen, velocidad, variedad, variabilidad, valor y complejidad” (Mohammad, 2013).

De esta manera el reto de mejorar el procesamiento de Big data se vuelve una tarea más complicada por los factores y condiciones a considerarse, pero necesaria en cualquier campo de la vida actual; no se debe tampoco, en ninguna circunstancia, menospreciar el costo que conlleva considerar todas y cada una de las propiedades o definiciones de Big data en su correcto tratamiento.

2.2 FLUJOS DE DATOS

No existe una definición clara o establecida para flujos de datos, pero se podría decir simplemente que son datos que se generan y/o se extraen de manera continua o secuencialmente y que “pudieran ser incorporadas en un mismo canal de transmisión” (Diván, 2011).

Para aclarar un poco el nacimiento y concepto inicial de los flujos de datos se encuentra que según el trabajo “Algoritmos de agrupación para flujos de datos en entornos centralizados y distribuidos” existe un paradigma antiguo llamado *store-then-process* donde, como su nombre lo dice, primero se almacenan los datos y después se consultan. El problema con este patrón es la “alta latencia para producir respuestas a las consultas. Por consecuencia no es posible satisfacer necesidades en procesamiento rápido” (Callau Zori, 2012), con esta problemática reconocida es que se piensa en los flujos como una posible solución.

En consecuencia, para satisfacer y solucionar los problemas anteriores nacen los flujos de datos (data stream) como un modelo o paradigma en donde se procesan los datos a su llegada mediante algoritmos. Los resultados se visualizan al tener respuestas continuas, un solo procesamiento cada vez, procesamiento complejo a la entrada para salidas bajas y espacio de memoria reducido en ciertos casos (Callau Zori, 2012).

En el mismo trabajo de Zori Callau, con los flujos de datos como modelo se entienden también dos enfoques: modelo evolutivo y modelo de ventana deslizante.

- Modelo Evolutivo. - se considera un histórico.
- Modelo de ventana deslizante. - se consideran datos recientes. (Callau Zori, 2012)

Además, se debe tomar en cuenta que cuando se presentan flujos de datos a éstos se los piensa como "transitorios", y por tanto las fuentes de su procedencia puede ser muy variada.

2.2.1 Tipos de flujos de datos

De acuerdo con el grupo de autores del artículo "Transmisión Eficiente de Bloques en Tiempo Real sobre Redes IP", desde el punto de vista de QoS que considera entre otros la disponibilidad, retardo, pérdidas de paquetes, etc.; en servicios de transmisión IP, los flujos de datos se pueden dividir en 3: "elásticos, semi-elásticos e inelásticos" (Postigo-Boix, Aguilar-Igartua, & García-Haro, 2014)

- **Elásticos.** - No requieren de alta calidad, y por tanto se tolera fallos en mayor escala a comparación de los otros.
- **Semi-elásticos**. - Se encuentra en la mitad, es decir su calidad es limitada, pero se necesita de la llegada de la información al usuario final. Aquí reposan aplicaciones en tiempo real por bloques.
- **Inelásticos.** - Requieren de alta calidad "fija e invariable". Aquí reposan las aplicaciones con flujos de datos en tiempo real muy sensibles a pérdidas o retardos. (Postigo-Boix, Aguilar-Igartua, & García-Haro, 2014)

2.2.2 Operadores de estado para flujos de datos

Los operadores de estado se consideran para comprender cómo se encuentra un flujo (en este caso se contemplarán tuplas) de datos en relación a su procesamiento. Sobre los flujos de datos se realizan operaciones, dependiendo de la necesidad y el tipo de operación que se requiere, se

dividen en dos los operadores de estado que se les puede otorgar: Stateless (sin estado) y Statefull (con estado).

- **Stateless (sin estado). -** En general aquí se procesa un registro a la vez y su característica principal es que este registro no depende de otro para crear información. Según lo expuesto en el trabajo de Zori Callau existen 3 operadores:
    - *Filter*: Salen tuplas que cumplen restricciones específicas.
    - *Map*: Salen tuplas después de haberse aplicado una o varias funciones a una única tupla de entrada
    - *Union*: En la salida se obtiene 2 o más streams unidos, es decir tuplas en secuencia.
- **Statefull (con estado).** – Para tener un flujo statefull de salida, se debe partir de un conjunto de flujos de entrada consecutivos; las operaciones que se pueden obtener son:
    - *Aggregate*: la tupla es resultado de una función aplicada anteriormente a otro conjunto de tuplas de entrada.
    - *Join*: "tienen como entrada dos flujos produciendo una tupla de salida para cada par de tuplas de entrada que satisfagan un predicado donde solo se miran un conjunto de tuplas de entrada consecutivas en cada *stream*".

2.2.3 Modelo de flujos de datos (The Data Stream Model)

Se debe entender o visualizar a los flujos de datos como transitorios, debido a que tienen una entrada única a memoria y de ahí que sigan como una fila y pasen una sola vez por dicha entrada. Según el trabajo "Models and Issues in Data Stream Systems", los flujos de datos difieren de lo que se entiende por convención en un modelo relacional con algunas de las siguientes características:

- Manera de llegada *on-line.*
- No se puede tener control sobre el orden de procesamiento de los datos, los flujos de datos o un solo flujo de dato.
- Tamaño potencialmente ilimitado.

- Al terminar de procesarse un elemento de un flujo de datos, éste automáticamente se descarta o archiva, generalmente no es fácil de almacenar en memoria debido a que su tamaño es pequeño en relación al tamaño del flujo de datos.
  (Babcock, Babu, Datar, Motwani, & Widom, 2002)

2.3 HERRAMIENTAS PARA BIG DATA

Actualmente existen muchas tecnologías y herramientas que se han desarrollado para solventar las complicaciones con los llamados "grandes volúmenes de datos". Así, se puede observar una evolución clara con respecto a las técnicas que desarrollan, y la diferencia entre una u otra herramienta es precisamente el modo en que realizan el procesamiento de los datos.

Debido a la naturaleza del proyecto, las herramientas escogidas para un análisis preliminar son aquellas enfocadas a procesamiento en streaming y de código abierto. Entre las más populares a la fecha se encontraron Hadoop (la más popular de las herramientas), Spark Streaming, Apache Storm, Apache Flink, Apache Kafka. A continuación, se ofrecerá un breve resumen de las herramientas escogidas preliminarmente.

2.3.1 Hadoop

Es una de las herramientas más conocidas para Big data. Precisamente por esta popularidad se encuentra mucha información sobre ella en la red y sobre la revolución que causó en un inicio con sus técnicas.

Esta herramienta es un proyecto de software abierto. Hadoop en sí es una biblioteca con un framework de código abierto, sus modelos de programación son mundialmente conocidos por ser simples para procesamiento y distribuidos. Su diseño está hecho para superar fallos en la capa de aplicaciones, ofreciendo una alta precisión. Es usada en la actualidad por gigantes del internet como Yahoo o Facebook. (Baoss Analytics Everywhere, 2015)

A pesar de todo lo mencionado anteriormente y dada la fortuna de Hadoop, muchas otras herramientas han complementado o mejorado sus capacidades dejándola atrás, pero no destronando del todo su liderazgo en el mercado. De cualquier manera, Hadoop, siempre será un referente en lo que a Big data respecta y por lo tanto la mayoría de herramientas mejoradas siguen partiendo de la primicia de compatibilidad o complemento con sus componentes.

2.3.2 Spark Streaming

Es un sistema de computación distribuida en clústers de computadores. Spark streaming tiene como base Apache Spark, y éste a su vez parte de los conceptos básicos de Hadoop como MapReduce; es decir que Apache Spark tiene a su disposición a DAG (Direct Acyclic Graph) y RDD (Resilent Distributed Dataset). Con estas dos funcionalidades a su disposición, Spark Streaming mejora la tolerancia a fallos de MapReduce y a la vez reduce costos de procesamientos al explotar al máximo la memoria caché y no escribir en disco cada parte del proceso.

Spark Streaming tiene la característica de poder recibir datos de otras fuentes "como Apache Kafka, Flume, Twitter, ZeroMQ, Kinesis o sockets TCP, los cuales pueden ser procesados utilizando complejos algoritmos a los que se accede mediante funciones de alto nivel como map, reduce o join" (Pérez Esteso , 2015).

Figura 2. Fuentes de almacenamiento de Spark Streaming

Fuente: (Apache Spark, 2017)

Una vez recibidos dichos datos el procesamiento se realiza por lotes (batchs), y es cuando se puede configurar el tiempo de procesamiento en cada lote y de cuántos datos consta cada uno. Debido a la naturaleza de procesamiento en lotes, siempre suele existir un grado de latencia y esto puede representar una desventaja para Spark Streaming.

2.3.3 Apache Storm

Esta herramienta es un sistema de computación distribuida de código abierto. Su mayor ventaja es el funcionamiento en tiempo real y el hecho de que el sistema se basa en construir topologías de Big data para transformar y analizar los datos que entran constantemente en un proceso continuo.

Una de las características es que "Apache Storm garantiza el procesamiento de un dato de entrada al menos una vez, lo que puede causar inconsistencia debido a que un dato de entrada puede ser procesado dos veces" (Pérez Esteso , 2015)

Pero de todos modos es una herramienta fiable y sencilla que maneja eventos complejos (Complex Event Processing, CEP) en un sistema de procesamiento. Lo ideal, desde la visión empresarial, es que Apache Storm permite responder de manera inmediata a eventos repentinos, y por ende ayuda en la toma de decisiones. Además, es interesante por algunas de las siguientes razones:

- Varios lenguajes de programación:
  - Storm está desarrollado en Clojure, un dialecto de Lisp que se ejecuta en Máquina Virtual Java (JVM, en sus siglas en inglés).
  - Compatibilidad con componentes y aplicaciones escritos en varios lenguajes como Java, C#, Python, Scala, Perl o PHP.
- Es escalable.
- Tolerante a fallos.
- Fácil de instalar y operar.

(BBVA, 2015)

2.3.4 Apache Flink

Flink es un motor de procesamiento de flujos de datos (stream). Se caracteriza por haber sido pensado como una alternativa a lo propuesto por Hadoop con MapReduce; es decir que se

presenta con un procesamiento distribuido y que a la vez acepta todo lo propuesto por Hadoop como HDFS o YARN (MapReduce v2). Apenas en febrero de este año (2017), Apache Flink lanzó su última versión 1.2.0 que es considerada finalmente como la versión más estable

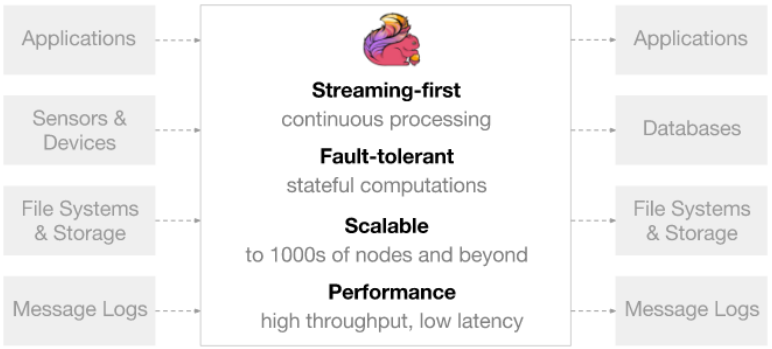


Figura 3. Características de Apache Flink

Fuente: (Apache Flink, 2016)

Apache Flink es un API con la capacidad de ser utilizado con otras fuentes de almacenamiento y cuya gran fortaleza es poder ejecutarse de dos maneras: streaming y por lotes, teniendo un procesamiento por evento y no necesariamente configurable en tiempo o velocidad.

Entre las características que sobresalen para Flink están:

- Funcionamiento a gran escala, podría ejecutarse en miles de nodos y mantener buen rendimiento, así como características de latencia.
- Al tener un estado y ser tolerante a fallos, la recuperación a los mismos es buena mientras mantiene el estado en cada aplicación con cero pérdidas.
- Entrega resultados precisos incluso cuando han existido desfases en la llegada de datos.

(Apache Flink, 2016)

2.3.5 Apache Kafka

Apache Kafka recientemente lanzó su API llamada Kafka Streaming en una versión más estable. Anteriormente esta plataforma no se veía como un procesador de datos en tiempos real;

sino más bien, se le consideraba en la parte baja de una arquitectura y necesitaba de otras herramientas como Apache Spark (o viceversa), es decir que servía únicamente como almacenamiento (para simbolizarlo de alguna manera).

Para la división de Apache Kafka una plataforma de transmisión tiene tres capacidades:

- Permite publicar y suscribirse a flujos de registros similar a una cola de mensajes o a un sistema de mensajería empresarial.
- Permite almacenar flujos de registros con tolerante a fallos.
- Permite procesar flujos de registros mientras ocurren.

Para lograr lo anterior, Kafka también recurre a eventos, es decir que pretende procesar los datos cuando éstos ocurren. Apache Kafka sirve para realizar dos tipos de aplicaciones:

- "Creación de tuberías de datos en tiempo real, obteniendo datos fiables entre sistemas o aplicaciones.
- Creación de aplicaciones de streaming en tiempo real que transforman o reaccionan con flujos de datos ".

(Apache Kafka, 2017)

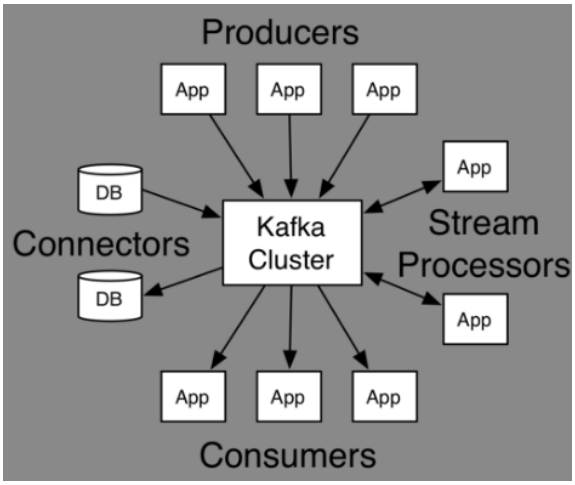


Figura 4. Core APIs de Apache Kafka

Fuente: (Apache Kafka, 2017)

Adicionalmente, como se muestra en la figura X, Kafka considera 4 tipos de APIs para su funcionamiento integral que son: API de Producción, API de Consumidor, API de Streams y API Conector. Con estas cuatro fortalezas a su favor, jamás pierde la capacidad de complementarse con otras herramientas, lo que le suma otra gran ventaja.

2.4 SCALA (SCALABLE LANGUAGE)

Según el sitio oficial, Scala es un lenguaje de programación moderno multi-paradigma que fue diseñado para que patrones de programación comunes se puedan expresar de manera concisa, elegante y segura (École Polytechnique Fédérale, 2017). La principal característica que hace de Scala un lenguaje de programación apto para todo y para todos es el hecho de combinar tanto un lenguaje orientado a objetos como un lenguaje funcional.

Además, y como un dato adicional introductorio, Scala es un lenguaje compilador que utiliza la máquina virtual de Java para su funcionamiento (JVM -Java Virtual Machine), por lo que asociar su funcionamiento y/o estructuración al lenguaje de Java no es descabellado.

2.4.1 Características de Scala

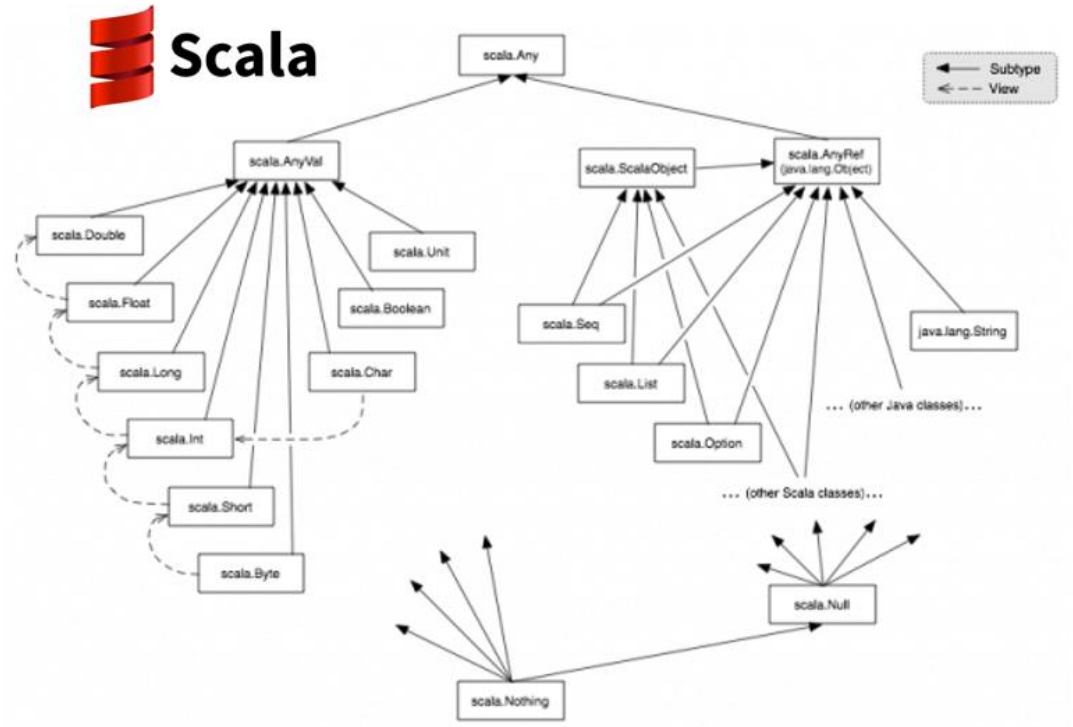


Figura 5. Jerarquía de clases en Scala

Fuente: (École Polytechnique Fédérale, 2017)

2.4.1.1 Multiparadigma

La característica más destacable de Scala, como se mencionó anteriormente es que puede desenvolverse tanto como lenguaje Orientado a Objetos como lenguaje Funcional. Todos sus tipos de unidades se derivan de dos grandes clases: *scala.AnyVal* y *scala.AnyRef*, clases para valores y clases para referencias correspondientemente.

- **scala.AnyVal:** Son las clases para valores predefinidas y correspondientes con los tipos de valores primitivos (y que también son predefinidos) en lenguaje Java.
- **scala.AnyRef:** Son tipos de unidades o clases referenciables. Estas clases serían, por ejemplo, las clases que crea el usuario y por defecto extenderían de esta clase *scala.AnyRef*.

2.4.1.2 Lenguaje compilador

Scala es un lenguaje compilado. Existen, en este sentido, dos tipos de lenguaje que son compilador e intérprete. Un lenguaje **intérprete** es aquel en el cual sus instrucciones o código fuente (que ha sido escrito por un programador en alto nivel) se debe interpretar por la máquina con un lenguaje entendible para sí misma instrucción por instrucción, todo ello cada vez que se ejecute el programa este proceso se repite. Mientras que, el lenguaje **compilado** simplemente se traduce para la máquina mediante un archivo ejecutable en determinada plataforma, y este proceso se realiza una sola vez sin importar cuántas veces sea ejecutado el programa.

A diferencia de otros lenguajes de compilado, Scala utiliza la máquina virtual de Java para su ejecución. Java como tal, es un lenguaje que se considera interpretado y compilado por su ventaja con su máquina virtual. La principal ventaja de esta máquina virtual es que puede ejecutar código Java en cualquier máquina donde esté instalado JVM.

Específicamente para Big data, el que Scala sea un lenguaje compilado ayuda a que la ejecución sea mucho más rápida, comparado por ejemplo con Python, un lenguaje compilado cuyo código es repetitivo de cierta manera, es más útil. Adicionalmente se debe mencionar que lenguajes puramente funcionales como Haskell, son lenguajes compilados.

2.4.1.3 Terminología básica de Scala

Además de los tipos unificados, Scala tiene ciertos términos básicos que son importantes de mencionar.

- **Objetos:** Entidades con un estado y comportamiento.
- **Clases:** Consideradas como una huella o modelo para crear objetos en las que se definen propiedades y comportamientos.
- **Métodos:** Son los diferentes comportamientos que pueden tener las clases.
- **Closure:** Es una función cercana al ambiente en el que está definido. Esta función devuelve una variable pero que depende de otras variables ajenas o externas a la función closure actual.
- **Traits:** Son los equivalentes a las interfaces en Java, es decir que definen comportamientos de acuerdo a determinados métodos. La diferencia con Java es que

en Scala se puede implementar los traits pero con limitaciones (se puede implementar procesos por defecto) y que no pueden tener constructores con parámetros.

2.5 METODOLOGÍA PARA PATRONES DE COMPORTAMIENTO: ISO/IEC 25000

Como metodología para revisar y determinar cuáles son los patrones de comportamiento de cada herramienta se va a utilizar la Norma ISO/IEC 25000 para Calidad del producto Software.

A pesar de la naturaleza de las herramientas para Big data, la ISO 25000 provee una fuente completa y adaptable para relacionar los comportamientos de las mismas y por tanto nos permitirá determinar cuál es la mejor para el tratamiento de los datos.

2.5.1 Generalidades

Según el mismo portal de la Norma ISO/IEC 25000, la calidad del producto software y la calidad del proceso son actualmente parte de los aspectos más importantes a considerar en el desarrollo de software; siendo el primero, de la calidad del producto, el que trata esta norma.

A su vez, se puede mencionar que esta norma es también una guía para los estándares internacionales de “Requisitos y Evaluación de Calidad de Productos de Software (SQuaRE - System and Software Quality Requirements and Evaluation)”.

El principal objetivo de la norma se centra en “guiar el desarrollo de los productos de software mediante la especificación de requisitos y evaluación de características de calidad” (International Standarization Organization ISO 25000, 2017)

La ISO 25000 nace en el año 2005 a partir de dos normas, la ISO/IEC 9126 (familia de las ISO 9000) y la ISO/IEC 14598 (a partir de la ISO 9126).

2.5.2 Norma ISO/IEC 9126

Este estándar es específico para medir la calidad de software y nació en el año 1991 como parte de la familia de las ISO 9000. En la actualidad está reemplazado por el ISO 25000,

Se divide en cuatro partes:

- **ISO 9126 – 1: 2001.** Estructura las características para medir la calidad: funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad.

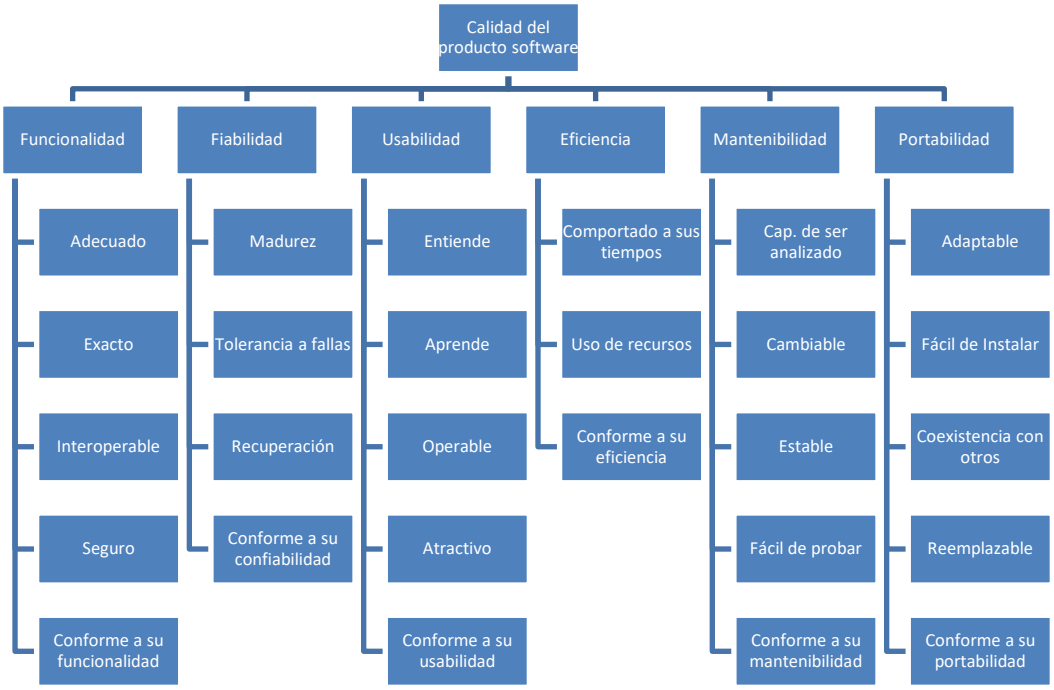


Figura 6. Características de Calidad Interna y Externa

Fuente: (International Standarization Organization ISO 9126, 1991)

Como se muestra en la Figura 6, cada característica tiene sus propios lineamientos básicos para evaluar la calidad en un sentido práctico y visible, es decir, un sentido que le permite al evaluador ver al software por partes y poder evaluarlas independientemente.

- **ISO 9126 - 2: 2003.** En esta parte de la norma se habla de las métricas externas con que se evalúa en el momento de que el software es operado por el usuario.

- **ISO 9126 – 3: 2003.** En esta parte de la norma se habla de las métricas internas con que se evalúan propiedades estáticas de la interfaz, esto sin necesariamente tener el software funcionando y poder hacerlo solo con inspecciones rápidas.
- **ISO 9126 – 4: 2004.** Se puede evaluar el efecto que produce un producto software cuando se somete en determinado contexto, es por eso que en esta sección se describen métricas de calidad para el efecto. Debe tomarse en cuenta que esto es directamente una evaluación para el usuario y que por lo tanto se evaluará lo siguiente:
    - **Efectividad (Test a usuarios). -** Efectividad de tareas (si se cumplen objetivos), tareas completadas, frecuencia de errores
    - **Productividad (Test a usuarios). -** Tiempo de tareas, efectividad de tareas, productividad económica, porcentaje de producción, eficiencia relativa de los usuarios.
    - **Seguridad (Estadísticas). –** Relación entre salud y seguridad del usuario, seguridad de quienes han visto afectada su salud debido al uso del producto, daño ergonómico, daño del software.
    - **Satisfacción (Test a usuarios). –** Escala de satisfacción general, cuestionario de satisfacción sobre características específicas del software, uso discrecional (¿qué proporción de usuarios usan el sistema?, usualmente se realiza por observación).

Como se puede observar, la ISO 9126 dictamina qué se debe/puede evaluar, e incluso muestra virtualmente una guía de cómo se debe evaluar a cada característica.

Sin embargo, al hablar de un producto software de manera íntegra, el ISO 9126 no se considera completo si se trata de evaluarlo de la misma manera íntegra, por tanto, se debe obligatoriamente aplicar también el ISO 14598.

2.5.3 Norma ISO/IEC 14598

La ISO 14598 es un estándar con un marco de trabajo que ayuda en la evaluación de la calidad del software, pero se centra también en los requisitos que son necesarios para la medición (métricas), de esta manera viene a ser un vínculo o un complemento de la norma ISO 9126. En general presenta tres situaciones diferentes:

- Requisitos para desarrolladores
- Requisitos para compradores
- Requisitos para evaluadores.

Tomando en cuenta estos tres requisitos, la norma tiene seis pasos o específicamente el proceso de evaluación consta de seis partes:

- **ISO/IEC 14598 – 1:** 1999 (Descripción General). Provee una explicación rápida entre la relación de sus siguientes partes con la norma ISO 9126 por tanto, muestra claramente su relación y por qué deben usarse en conjunto.
- **ISO/IEC 14598 – 2:** 2000 (Planificación y administración).
- **ISO/IEC 14598 – 3:** 2000 (Proceso para desarrolladores).
- **ISO/IEC 14598 – 4:** 1999 (Proceso para adquisidores).
- **ISO/IEC 14598 – 5:** 1998 (Proceso para evaluadores).
- **ISO/IEC 14598 – 6:** 2001 (Documentación Módulos de evaluación).

2.5.4 Norma ISO/IEC 25000

La ISO/IEC 25000 es una familia de normas cuyo principal objetivo es crear “un marco de trabajo común para evaluar la calidad del producto software” (International Standarization Organization ISO 25000, 2017). Las divisiones de esta familia se muestran en la Figura X, formando un cuadrado para medición y evaluación de la calidad, de ahí que algunas veces únicamente se lo llame por el acrónimo *SQuaRE*, como se mencionó en las generalidades.

Figura 7. SQuaRE (System and Software Quality Requirements and Evaluation)

Fuente: (International Standarization Organization ISO 25000, 2017)

En la Tabla 1 se encuentra un breve resumen sobre cada una de sus partes:

Tabla 1

Resumen general de contenido del estándar ISO 25000

ISO/IEC	NOMBRE	DESCRIPCIÓN	DIVISIONES
2500n	División de Gestión de Calidad	Define modelos, términos y definiciones comunes a todas las familias.	**25000:** Guide to SQuaRE. **25001:** Planning and Management.
2501n	División de Modelo de Calidad	Modelos de calidad detallados con características de calidad interna, externa y en uso del producto software.	**25010:** System and software quality models. **25012:** Data Quality model.
2502n	División para la Medición de Calidad	Modelo de referencia para medir la calidad del producto software, definición de medidas de calidad, y guías y prácticas para su aplicación.	**25020:** Measurement reference model and guide. **25021:** Quality measure elements. **25022:** Measurement of quality in use. **25023:** Measurement of system and software product quality. **25024:** Measurement of data quality.

CONTINÚA

2503n	**División para los Requisitos de Calidad**	**Requisitos de calidad para el proceso de elicitación de requisitos o como entrada del proceso de evaluación.**	25030: **Quality requirements.**
2504n	División para la Evaluación de Calidad	Contiene requisitos, recomendaciones y guías para el proceso de evaluación del producto software.	**25040:** Evaluation reference model and guide. **25041:** Evaluation guide for developers, acquirers and independent evaluators. **25042:** Evaluation modules. **25045:** Evaluation module for recoverability

CAPÍTULO III

DESARROLLO COMPARATIVO

3.1 DETERMINACIÓN DE TÉCNICAS T1 Y T2

En este punto se ha visto la necesidad de hacer un análisis comparativo entre las herramientas populares para Big data que se describieron en el Capítulo II de Herramientas para Big data.

Se han tomado en consideración puntos esenciales específicos para la naturaleza de esta investigación, y a continuación en la Tabla 2, se presentan dichos puntos. Con el desarrollo de este análisis preliminar se pretende descubrir cuáles son las dos mejores opciones para realizar un estudio de comportamiento más a fondo.

Es decir que con el contenido de la Tabla 2 se predefinirán los puntos sustanciales para realizar la comparación entre las 5 herramientas potenciales, dos de las cuales se someterán al análisis según la Norma ISO 25000 posteriormente. La tabla muestra a breves rasgos lo más importante encontrado en la investigación anterior para esta depuración y es de elaboración propia.

Tabla 2

Análisis comparativo de herramientas de Big data

	HADOOP	**SPARK STREAMING**	**APACHE STORM**	**APACHE FLINK**	**APACHE KAFKA**
Año de creación	2006	2009	2014	2015	2017 (2011)
Desarrollador inicial	Yahoo	UC Berkeley AMP Lab	Twitter	Apache	LinkedIn
Tecnología de procesamiento de datos	Batches (por lotes)	Batches (por lotes)	Tiempo Real	Tiempo Real	Tiempo Real
Latencia	Alta	Muy baja	Media Baja	Baja	Baja
Tolerancia a fallos	Muy estable (en memoria).	Estable (en clúster distribuidos)	Estable (detecta fallas a nivel de tareas)	Estable (Con complemento de almacenamiento)	Muy estable (debido a su API de Stream)
Escalabilidad	Lineal	Lineal	Lineal	Horizontal	Horizontal
Lenguaje de programación	Java / C++ / Python	Scala / Java / Python	Scala / Java / Python / Ruby / Perl / C# / PHP	Scala / Java / Python	Scala / Java
Principal característica	Modelo de programación MapReduce con HDFS (Hadoop Distributed File System)	Procesamiento de datos por lotes y a velocidad configurable. Modelo de procesamiento RDD (Resilent Distributed Datasets).	Procesamiento de datos en tiempo real y con una distribución parecida a MapReduce, pero mejorada.	Potente procesador de datos, pero necesita de un complemento para el almacenamiento de datos (Apache Kafka, etc.).	Apache Kafka tiene a su disposición el API Kafka Stream que permite procesar datos en cuanto ocurren (eventos).

3.2 SPARK STREAMING (T1) VS. APACHE FLINK (T2)

A la luz de esta última investigación y su análisis presentado en la Tabla 2, se consiguió finalmente extraer dos herramientas que se consideran idóneas para el flujo de datos. Las razones que pesaron más para ser escogidas las herramientas son:

1. **Calidad y cantidad de documentación**: Debido a que entre las herramientas consideradas están algunas relativamente nuevas, se torna de mucha importancia que éstas tengan la documentación suficiente para que sean comparables y verificadas oficialmente.
2. **Procesamiento en tiempo real:** Una de las principales características que debía poseer la herramienta era el procesamiento en tiempo real. Estas herramientas nos permiten comparar las diferencias que representaría uno u otro modo de procesamiento (tiempo real o por lotes).
3. **Lenguaje de programación:** La ventaja de ciertas herramientas es que permiten programación en lenguaje Scala, que cambia el paradigma de programación a funcional, y no únicamente la tradicional orientada a objetos. Además, la tendencia en herramientas para Big data en general se ha inclinado por programación funcional, apuntando a actualizar y mejorar los procesos eficientemente.

Lo mencionado anteriormente son los tres factores que más influenciaron para escoger las herramientas. Brevemente se puede mencionar que herramientas como Apache Storm o Hadoop no se escogieron debido a presentar menos avances o progresos para data stream en los últimos tiempos; y que Apache Kafka no se escogió por apenas haber lanzado la primera versión estable en agosto de 2017 y no presentar cambios desde dicha fecha, por esto es cuestionable la cantidad de documentación y fiabilidad que pueda presentar esta herramienta poco explorada todavía.

Así, se afirma el tener como competidoras a, las que desde ahora se denominarán herramientas T1 y T2 respectivamente, SPARK STREAMING y APACHE FLINK como las herramientas a comparar.

3.3 ESTUDIO COMPARATIVO: ISO 25000

Según el análisis realizado a la norma internacional ISO 25000 se construirá un modelo para la evaluación de las herramientas T1 y T2.

3.3.1 Métricas para evaluación

Para las métricas de evaluación se establecen dos tipos fundamentales: Métricas por rango y Métricas de cumplimiento.

3.3.1.1 Métricas por rango

Dependiendo cómo sea el comportamiento de la herramienta T1 comparado con la herramienta T2, se otorgará una calificación en un rango ascendente entre 0 y 4. La tabla 3 muestra la descripción de cada calificación.

Tabla 3

Métricas por Rango

VALOR	DESCRIPCIÓN
0	MUY MALO
1	MALO
2	REGULAR
3	BUENO
4	MUY BUENO

3.3.1.2 Métricas de cumplimiento

Para las métricas de cumplimiento se tiene como base un enfoque diferente a las métricas por rango. Simplemente se da un valor de cumplimiento o no cumplimiento, estos valores se han definido entre 0 y 1.

Tabla 4

Métricas por Cumplimiento

VALOR	DESCRIPCIÓN
0	CUMPLE
1	NO CUMPLE

3.3.2 Características de calidad

El modelo de evaluación tiene algunas características y sub características que nos ayudan a discernir de mejor manera qué evaluar. Las características son:

Tabla 5

Características y sub características del modelo de evaluación

TIPO	CARACTERÍSTICA	SUB CARACTERÍSTICA
Calidad Interna y Externa	Funcionalidad	Adecuación
		Exactitud
		Interoperabilidad
		Seguridad de acceso
	Fiabilidad	Madurez
		Tolerancia a Fallos
		Capacidad de Recuperación
	Usabilidad	Capacidad para ser Entendido
		Capacidad para ser Aprendido
		Capacidad para ser Operado
		Capacidad de Atracción
	Mantenibilidad	Capacidad para ser Analizado
		Capacidad para ser Probado
		Capacidad para ser Cambiado
		Estabilidad
	Portabilidad	Adaptabilidad
		Facilidad de Instalación
		Coexistencia
		Capacidad de reemplazo
	Eficiencia	Comportamiento Temporal
		Utilización de recursos
Calidad de Uso		Productividad
		Seguridad
		Satisfacción

Cabe recalcar que los tipos y características son dados por la norma ISO 25000 por tanto no se han alterado. A su vez cada sub característica puede tener tantas descripciones como sea

necesario, dichas descripciones mencionan puntualmente qué se va a evaluar para este caso comparativo.

De esta manera se ha construido el siguiente cuadro con las descripciones a usar para evaluar el comportamiento de las herramientas

3.3.3 Descripción de características del modelo

En la tabla 6 se muestra qué específicamente se va a evaluar en cada sub característica presentada en la NORMA ISO 25000. De esta manera queda claro qué partes son las que corresponden a la evaluación de las herramientas.

Tabla 6

Descripción de características y sub características del modelo de evaluación

TIPO	CARACTERÍSTICA	SUB CARACTERÍSTICA	DESCRIPCIÓN
Calidad Interna y Externa	Funcionalidad	Adecuación	Tiene los componentes necesarios para registros (logs)
		Exactitud	Concordancia entre procesamiento de datos y resultados
		Interoperabilidad	Mantiene compatibilidad con otros software o herramientas necesarias
		Seguridad de acceso	Cuenta con licencias libres
	Fiabilidad	Madurez	Tiempo en el mercado
		Tolerancia a Fallos	Módulos para manejo de excepciones o fallas.
		Capacidad de Recuperación	Módulos o mecanismos para reestablecer desempeño, además backup y restore.
	Usabilidad	Capacidad para ser Entendido	Documentación suficiente con ejemplos teóricos y prácticos.
		Capacidad para ser Aprendido	
		Capacidad para ser Operado	N/A
		Capacidad de Atracción	N/A
	Mantenibilidad	Capacidad para ser Analizado	Soluciones en la comunidad y sitio oficial
		Capacidad para ser Probado	
		Capacidad para ser Cambiado	Cuenta con una compatibilidad adecuada entre versiones
		Estabilidad	Soporte adecuado y respaldo de la comunidad

CONTINÚA

	Portabilidad	**Adaptabilidad**	**Capacidad para moverse y adaptarse al nuevo mercado**
		Facilidad de Instalación	Descarga desde sitio Oficial, con soporte de instalación
		Coexistencia	N/A
		Capacidad de reemplazo	N/A
	Eficiencia	Comportamiento Temporal	Tiempos de respuesta
		Utilización de recursos	Cantidad de recursos utilizados en la máquina
Calidad de Uso		Productividad	Adecuado para intereses particulares (Procesamiento en tiempo real)
		Seguridad	Parches, actualizaciones y mejoras de seguridad
		Satisfacción	N/A

Para terminar con la elaboración del modelo, se ha especificado qué tipo de métrica se aplicará sobre cada sub característica. Como se muestra en la Tabla 7, las métricas para este modelo de evaluación quedan de la siguiente manera.

3.3.4 Métricas y estructura para modelo de calidad

Tabla 7

Métricas del modelo de calidad para el modelo actual

TIPO	CARACTERÍSTICA	SUB CARACTERÍSTICA	DESCRIPCIÓN	PONDERACIÓN MÁXIMA	HERRAMIENTAS	
					EVALUACIÓN	PUNTAJE
Calidad Interna y Externa	Funcionalidad	Adecuación	Tiene los componentes necesarios para registros (logs)	1	C	0-1
		Exactitud	Concordancia entre procesamiento de datos y resultados	4	R	0-4
		Interoperabilidad	Mantiene compatibilidad con otros software o herramientas necesarias	1	C	0-1
		Seguridad de acceso	Cuenta con licencias libres	1	C	0-1
	Fiabilidad	Madurez	Tiempo en el mercado	4	R	0-4
		Tolerancia a Fallos	Módulos de manejo de excepciones o fallas.	4	R	0-4
		Capacidad de Recuperación	Módulos o mecanismos para reestablecer desempeño, además backup y restore.	4	R	0-4
	Usabilidad	Capacidad para ser Entendido	Documentación suficiente con ejemplos teóricos y prácticos.	4	R	0-4
		Capacidad para ser Aprendido				
		Capacidad para ser Operado	N/A	N/A	N/A	N/A
		Capacidad de Atracción	N/A	N/A	N/A	N/A
	Mantenibilidad	Capacidad para ser Analizado	Soluciones en la comunidad y sitio oficial	4	R	0-4
		Capacidad para ser Probado				
		Capacidad para ser Cambiado	Cuenta con una compatibilidad adecuada entre versiones	1	C	0-1
		Estabilidad	Soporte adecuado y respaldo de la comunidad	4	R	0-4

CONTINÚA

	Portabilidad	**Adaptabilidad**	**Capacidad para moverse y adaptarse al nuevo mercado**	**4**	**R**	**0-4**
		Facilidad de Instalación	Descarga desde sitio oficial, con soporte de instalación	1	C	0-1
		Coexistencia	N/A	N/A	N/A	N/A
		Capacidad de reemplazo	N/A	N/A	N/A	N/A
	Eficiencia	Comportamiento Temporal	Tiempos de respuesta	4	R	0-4
		Utilización de recursos	Cantidad de recursos utilizados en la máquina	4	R	0-4
Calidad de Uso		Productividad	Adecuado para intereses particulares (Procesamiento en tiempo real)	1	C	0-1
		Seguridad	Parches, actualizaciones y mejoras de seguridad	4	R	0-4
		Satisfacción	N/A	N/A	N/A	N/A
			Total	50		

3.4 APLICACIÓN MODELO DE CALIDAD ISO 25000

En la aplicación del modelo de calidad se obtuvieron los siguientes resultados:

Tabla 8

Aplicación del modelo de calidad

SUB CARACTERÍSTICA	DESCRIPCIÓN	PUNTAJE MÁXIMO	HERRAMIENTAS	
			T1 (SPARK)	T2 (FLINK)
Adecuación	Tiene los componentes necesarios para registros (logs)	1	1	1
Exactitud	Concordancia entre procesamiento de datos y resultados	4	4	3
Interoperabilidad	Mantiene compatibilidad con otros software o herramientas necesarias	1	1	1
Seguridad de acceso	Cuenta con licencias libres	1	1	1
Madurez	Tiempo en el mercado	4	4	3
Tolerancia a Fallos	Módulos para manejo de excepciones o fallas.	4	4	4
Capacidad de Recuperación	Módulos o mecanismos para reestablecer desempeño, además backup y restore.	4	4	4
Capacidad para ser Entendido	Documentación suficiente con ejemplos teóricos y prácticos.	4	4	3
Capacidad para ser Aprendido				
Capacidad para ser Analizado	Soluciones en la comunidad y sitio oficial	4	4	3
Capacidad para ser Probado				
Capacidad para ser Cambiado	Cuenta con una compatibilidad adecuada entre versiones	1	1	1
Estabilidad	Soporte adecuado y respaldo de la comunidad	4	4	2
Adaptabilidad	Capacidad para moverse y adaptarse al nuevo mercado	4	4	2
Facilidad de Instalación	Descarga desde sitio Oficial, con soporte de instalación	1	1	1
Comportamiento Temporal	Tiempos de respuesta	4	3	3
Utilización de recursos	Cantidad de recursos utilizados en la máquina	4	3	3
Productividad	Adecuado para intereses particulares (Procesamiento en tiempo real)	1	1	1
Seguridad	Parches, actualizaciones y mejoras de seguridad	1	1	1
	TOTAL	**47**	**45**	**37**

3.4.1 Resultados de la aplicación del modelo de calidad

Tabla 9

Resultado integral de la aplicación del modelo de calidad

TIPO	CARACTERÍSTICA	SUB CARACTERÍSTICA	DESCRIPCIÓN	PUNTAJE MÁXIMO	TIPO MÉTRICA	HERRAMIENTA	
						T1 SPARK	T2 FLINK
Calidad Interna y Externa	Funcionalidad	Adecuación	Tiene componentes necesarios para registros (logs)	1	C	1	1
		Exactitud	Concordancia entre procesamiento de datos y resultados	4	R	4	3
		Interoperabilidad	Mantiene compatibilidad con otros software o herramientas.	1	C	1	1
		Seguridad de acceso	Cuenta con licencias libres	1	C	1	1
	Fiabilidad	Madurez	Tiempo en el mercado	4	R	4	3
		Tolerancia a Fallos	Módulos para manejo de excepciones o fallas.	4	R	4	4
		Capacidad de Recuperación	Módulos o mecanismos para reestablecer desempeño, además backup y restore.	4	R	4	4
	Usabilidad	Capacidad para ser Entendido	Documentación suficiente con ejemplos teóricos y prácticos.	4	R	3	3
		Capacidad para ser Aprendido					
	Mantenibilidad	Capacidad para ser Analizado	Soluciones en la comunidad y sitio oficial	4	R	3	3
		Capacidad para ser Probado					
		Capacidad para ser Cambiado	Cuenta con una compatibilidad adecuada entre versiones	1	C	1	1
		Estabilidad	Soporte adecuado y respaldo de la comunidad	4	R	4	3

CONTINÚA

	Portabilidad	**Adaptabilidad**	**Capacidad para moverse y adaptarse al nuevo mercado**	**4**	**R**	**4**	**3**
		Facilidad de Instalación	Descarga desde sitio Oficial, con soporte de instalación	1	C	1	1
	Eficiencia	Comportamiento Temporal	Tiempos de respuesta	4	R	4	2
		Utilización de recursos	Cantidad de recursos utilizados en la máquina	4	R	2	4
Calidad de Uso		Productividad	Adecuado para intereses particulares (Procesamiento en tiempo real)	1	C	1	1
		Seguridad	Parches, actualizaciones y mejoras de seguridad	4	R	4	3
		Total		**50**		**46**	**41**

3.4.2 Análisis de resultado

Según se puede observar en la Tabla 9, el resultado total señala como superior a la herramienta T1: Spark Streaming con respecto a la herramienta T2: Apache Flink, por una diferencia de apenas 5 puntos.

Cabe mencionar que ambas herramientas en casi todas las características se encuentran muy parejas. A continuación, se va a realizar un análisis más específico únicamente de 6 puntos en los cuales se ha encontrado disparidad y se explicará por qué se suscita la misma.

1. **Funcionalidad – Exactitud – Concordancia entre procesos de datos y resultados:** La diferencia clara se aprecia en la técnica de procesamiento. Spark procesa mediante batchs y Flink hace un procesamiento nativo o directo al ingreso del stream.
2. **Característica Fiabilidad – Madurez:** Spark lleva en el mercado desde el año 2009 y Flink desde 2015. A pesar de que ambas herramientas llevan un tiempo prudente en el mercado, es innegable que Spark ha ganado más adeptos con el tiempo y puede ser que dicha estancia prolongada en el mercado juegue en contra de herramientas relativamente nuevas como Flink.
3. **Mantenibilidad – Estabilidad y Portabilidad - Adaptabilidad:** En este punto no se ha tomado en cuenta versiones de las herramientas, sino en general la participación de la comunidad, así como la documentación presentada en diferentes fuentes oficiales y no oficiales. Este resultado en el que Spark es superior deriva directamente de la popularidad que tiene en todo el mundo. Además del hecho de que empresas tan grandes como LinkedIn la usan abiertamente para sus análisis.
4. **Eficiencia – Tiempos de Respuesta:** Según un experimento llevado a cabo por investigadores de Gradiant en 2017, Spark era superior a Flink con respecto a múltiples eventos en distintas ventanas debido al procesamiento en batch. Spark reúne toda la información en un tiempo establecido y lo procesa todo al instante, en cambio Flink no espera y procesa en el momento que llega cada dato. En los resultados ellos mencionan el tiempo de procesamiento en Flink que fue 869,3 segundos y en Spark 614,1 segundos.

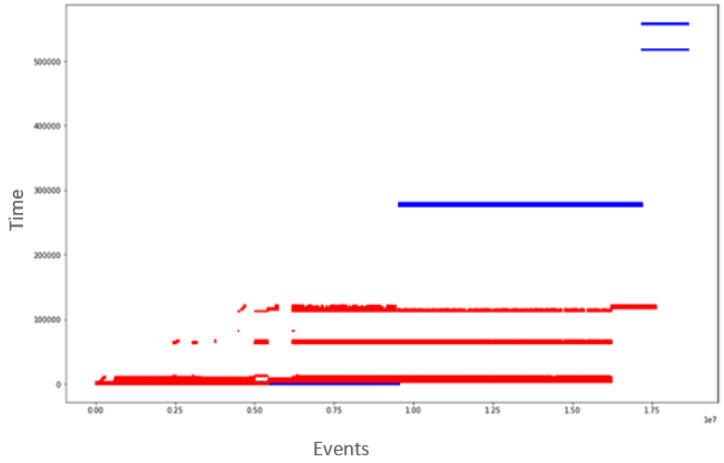


Figura 8. Eventos vs. Tiempo en caso de uso de proyecto Gradiant.
Fuente: (Portabales Goberna & Martínez Álvarez, 2017)

5. **Eficiencia – Cantidad de recursos utilizados por la máquina:** Continuando con el estudio anterior, Flink mostró ser superior al consumir menos de recursos. El hecho de que en tiempo de ingreso de datos Spark supere ampliamente a Flink, hace que consecuentemente utilice más recursos en el momento de hacer el cómputo, mas no en la ingesta de los datos.

   A pesar de lo obtenido por los investigadores de Gradiant en 2017, no se ha tenido en cuenta las mejoras en las versiones actuales de Spark en donde, por ejemplo, Spark introdujo datos estructurados con lo que se acerca al manejo estructurado de Flink.
6. **Seguridad:** En este punto Spark tiene más experiencia y por tanto mejoras superiores a Flink. Actualmente Spark está experimentando e introduciendo nuevas características que necesitan aún tiempo para ser probadas, y como en todo software nuevo, ser corregidas.

Para finalizar, y a la luz de los resultados anteriores, se procederá a realizar una solución utilizando la herramienta T1: SPARK STREAMING.

CAPÍTULO IV
DESARROLLO DE LA SOLUCIÓN

4.1 INTRODUCCIÓN

Según lo visto en los resultados del capítulo anterior, se procederá a realizar una solución con la herramienta ganadora SPARK STREAMING para procesamiento de flujos de datos continuos de invernadero.

En el invernadero de rosas ubicado en el IASA I, existe actualmente implementada una red de sensores distribuidos estratégicamente para medir continuamente las condiciones en las que están desarrollándose las flores. Dichas rosas están dentro del invernadero precisamente por la necesidad de un entorno controlado para su correcto crecimiento. Entre los sensores que se encuentran actualmente en funcionamiento, uno de los más importantes es el sensor de temperatura.

Para las rosas, la temperatura es uno de los factores más importantes dentro de su crecimiento puesto que puede tener un impacto directo en la presentación final que éstas muestran al momento de ser cosechadas y vendidas. Tanto el exceso de calor o de frío, son determinantes para la aparición de diversas enfermedades o para la limitación de un correcto crecimiento, que afectan el volumen de producción y cosecha.

Además, con la correcta monitorización de la temperatura, se pueden inferir otros factores que afectan a la rosa, por ejemplo, si la humedad en el ambiente es la correcta. La solución propuesta aquí, presenta a Big data como una alternativa que permite monitorizar anormalidades en la temperatura del invernadero con mayor eficiencia y efectividad.

4.2 REQUISITOS

Del invernadero de rosas ubicado en el IASA I, el sensor de temperatura se encuentra enviando datos cada 10 minutos todos los días. Dichos datos se almacenan tanto en una base de datos como en archivos.

A continuación, se muestran los requisitos funcionales basándose en las normas generales para cuidado de rosas en estado de crecimiento en invernaderos dictados por Expoflores (Asociación de Productores y Exportadores de Flores, 2017):

Tabla 10

Requisitos funcionales

CÓDIGO	REQUERIMIENTO	DESCRIPCIÓN
RF01	Procesar archivos inmediatamente.	Los archivos se incrementarán cada cierto tiempo no especificado, por tanto, todos los datos que se almacenan deben ser procesados en el momento de arribo, sin importar hora de llegada o rango específico de tiempo para llegada.
RF02	Filtrar los datos según hora y temperatura correspondiente a la etapa de crecimiento.	Los datos se consideran correctos con el siguiente filtro: • Horas de luz entre las 6h00 y 18h00 -> Temperatura entre: 20°C y 25°C. • Horas de oscuridad entre las 00h00 - 5h59 y 18h01 – 23h59 -> Temperatura entre 10° y 12°C. Todos los demás datos se consideran erróneos para el periodo de crecimiento de las rosas.
RF03	Agregar un discriminante a datos según el filtro.	Colocar un dato extra que diferencie los datos filtrados: • Temperatura y horas que correspondan con la descripción del requerimiento RF02: "**CORRECTO**" • Datos que no correspondan con el requerimiento RF02: "**INCORRECTO**"
RF04	Guardar en archivos los resultados.	Guardar datos filtrados en nuevos archivos que contendrán la siguiente información: • Id • Nodo • Fecha • Hora • Temperatura • Límite superior de temperatura • Límite inferior de temperatura • Filtro

4.3 DESARROLLO DE LA SOLUCIÓN

A continuación, se procederá a mostrar el desarrollo de la solución con SPARK STREAMING para el procesamiento de la data.

4.3.1 Especificaciones técnicas

Debe mencionarse algunas técnicas dispuestas en el entorno antes de proceder con el desarrollo de la solución:

- **Sistema Operativo:** Ubuntu16.04
- **Herramienta de desarrollo (IDE):** Eclipse 12.06
- **Herramienta de Big data:** Spark 2.2.1
- **Máquina Virtual de Java (JVM):** 8
- **Lenguaje de programación:** Scala

4.3.2 Estado de datos para pruebas antes del procesamiento

En orden de realizar las pruebas pertinentes, se tiene alrededor de **4278 datos para las mismas por cada uno de los sensores** obtenidos durante el periodo **del 18 de febrero de 2018 hasta el 17 de marzo de 2018**. Recordando que dichos datos son únicamente para realizar pruebas para esta solución; en producción el comportamiento, tanto de la herramienta como de los flujos de datos, es muy similar.

Los datos están almacenados en archivos con extensión .csv. Dichos archivos se encontrarán en una dirección específica dentro de una carpeta. En cada archivo se visualiza la siguiente información: **Identificador único, nodo, fecha, hora, temperatura, humedad, brillo, factor UV y calidad de aire**, como se muestra en la Tabla 11.

Tabla 11

Ejemplo de muestra de datos de archivos antes de ser procesados

ID	Nodo	Mes_dia	Hora	Temp	Humedad	Brillo	Factor_uv	Calidad_Aire
427	1	4/3/2018	17:37:41	14.3	72.3	74	1	39
428	1	4/3/2018	17:47:42	14.8	73.1	64	2	39
429	1	4/3/2018	17:57:42	15.3	70.9	71	1	38
430	1	4/3/2018	18:07:43	14.1	74.7	107	1	38
431	1	4/3/2018	18:17:43	13.6	77.9	133	1	39

La dirección de la carpeta donde se encuentran los archivos a ser procesados debe ser especificada y en lo posible no debe cambiar, puesto que cuando se comience el proceso de streaming la herramienta buscará en dicha dirección nuevos datos.

4.3.3 Comportamiento de los datos

Existen unas cuantas consideraciones que se deben hacer sobre el comportamiento que tienen los flujos de datos que llegarán a procesarse en la herramienta de Spark. Entre las consideraciones están tanto el modelo de datos que se maneja en el invernadero, así como también el tipo de datos que se están obteniendo.

4.3.3.1 Modelo de datos

Recordando lo que dice la teoría, los flujos de datos tienen dos modelos: evolutivo y de ventana deslizante. Actualmente, los datos que proceden para el análisis en esta solución siguen un modelo evolutivo, puesto que se requerirá de un archivo para su procesamiento.

Sin embargo, de ser necesario seguir un modelo de ventana deslizante, el código debe acoplarse simplemente en la creación del punto de entrada de los datos y no existiría problema alguno.

4.3.3.2 Tipo de flujo de datos

El stream o flujo de datos que se obtiene para el análisis es de tipo **Semi-elástico**, puesto que la información obtenida después del procesamiento debe llegar a manos del usuario final, pero se puede tolerar fallos.

El hecho de que los datos sean de tipo semi elástico, hace que sean idóneos para ser tratados con Spark streaming, debido a su naturaleza, pueden ser tratados mediante lotes o batchs. Adicionalmente, se considera que se puede tolerar cierto tipo de fallos en la naturaleza misma de los datos porque, a ojos del usuario final, se debe considerar, por ejemplo, la época del año en la que se encuentre la etapa de crecimiento para realizar ajustes, tanto en el invernadero como en el procesamiento.

4.3.4 Código para procesamiento de datos

El código elaborado para cumplir los requisitos especificados con el uso de Spark Streaming es realmente corto. Desde la versión 2.0 en adelante, ya se introduce como innovación el uso de datos estructurados, volviendo de esta manera más fácil la codificación y se vuelca con un mejor procesamiento a cada avance de versionamiento, incluso siguen dando soporte a versiones antiguas y mejorando sus respectivas características.

A continuación, se procederá a mostrar el código realizado para la solución con los datos en invernaderos.

- Los datos de los límites en cuanto a horas de luz/oscuridad y temperaturas respectivamente, se almacenaron en un archivo de la siguiente manera:

Tabla 12

Límites de horas de luz/noche y temperaturas correspondientes

hora_luz	hora_noche	lim_inf_dia	lim_sup_dia	lim_inf_noc	lim_sup_noc
6	18	20	25	10	12

El código de la Figura 9 contiene básicamente la lectura de los límites desde un archivo en una localización específica, la construcción de un Array de dos dimensiones para los datos como se muestra en la tabla 12 y finalmente se nombra cada dato con una variable específica para hacerlo más entendible posteriormente.

```
val lineas = Source.fromFile("/home/dalia/curso2/InvernaderoSpark/limites.csv")

val nrows = 2
val ncols = 6
val rows = Array.ofDim[String](nrows, ncols)
var count = 0
for (line <- lineas.getLines) {
 rows(count) = line.split(",").map(_.trim)
 count += 1
}
val hora_luz = rows(1)(0).toString()
val hora_noche = rows(1)(1).toString()
val lim_inf_dia = rows(1)(2).toString()
val lim_sup_dia = rows(1)(3).toString()
val lim_inf_noc = rows(1)(4).toString()
val lim_sup_noc = rows(1)(5).toString()
```

Figura 9. Código para lectura de límites de horas y temperaturas correspondientes

Para la lectura del stream se muestra el código según la Figura 10, en éste se crea una sesión de streaming de Spark, se coloca un nombre a la sesión, se da a entender cuál es el nodo principal, dónde se encuentra la carpeta de recuperación y finalmente se construye el punto de entrada del stream.

```
val spark = SparkSession
  .builder
  .appName("Alertas")
  .master("local[*]")
  .config("spark.sql.streaming.checkpointLocation", "/home/dalia/checkpoint/")
  .getOrCreate()
```

Figura 10. Creación del punto de entrada para iniciar la lectura del stream de datos

Como se mencionó en un inicio, desde la versión 2.0+, Spark ya trabaja con datos estructurados, para esto se necesita importar una librería (que en Scala puede realizarse en cualquier parte del código), y crear la estructura que va a tener el stream que se obtiene. Al ver el código de la Figura 11, automáticamente nos damos cuenta los datos que ingresarían por el punto de entrada claramente diferenciados.

```
import spark.implicits._

val tempSchema = new StructType()
 .add("id", IntegerType)
 .add("nodo", IntegerType)
 .add("mes_dia", StringType)
 .add("hora", StringType)
 .add("temperatura", FloatType)
```

Figura 11. Creación de estructura de datos del flujo de entrada

La sesión de Spark es una entrada para, al momento que ingresan los datos estos sean transformados automáticamente en DataFrames (API propia de Spark), que se asemejan a una colección, y con la que se puede trabajar para realizar transformaciones y los procesos necesarios a aplicarse sobre los datos.

La lectura del flujo de datos se observa en la Figura 12, aquí se especifica el esquema (Figura 11) que se usa para los datos estructurados, el formato de los archivos de donde proceden los datos, en este caso archivos .csv y finalmente la dirección de la carga de donde Spark automáticamente buscará nueva data para procesarse. Cabe recalcar que la lectura, como se mencionó antes queda a modo de colección como un DataFrame.

```
val df = (spark
 .readStream
 .schema(tempSchema)
 .format("csv")
 .load("/home/dalia/curso2/InvernaderoSpark/tem/"))
```

Figura 12. Entrada de flujo de datos(stream) en un DataFrame

La variable *df*, es básicamente un lote(batch) de datos que está listo para ser procesado. En esta versión de Spark, al soportar datos estructurados, es posible procesar los datos mediante el uso de sentencias SQL. Para que dichas sentencias tengan efecto se debe crear una vista temporal como en una base de datos relacional.

En la Figura 13, se observa claramente en el código la creación de esta vista temporal a la que se le asigna un nombre, en este caso “flor”. Por tanto, esta vista temporal sería como la tabla sobre la cual se aplican los filtros y la sentencia SQL.

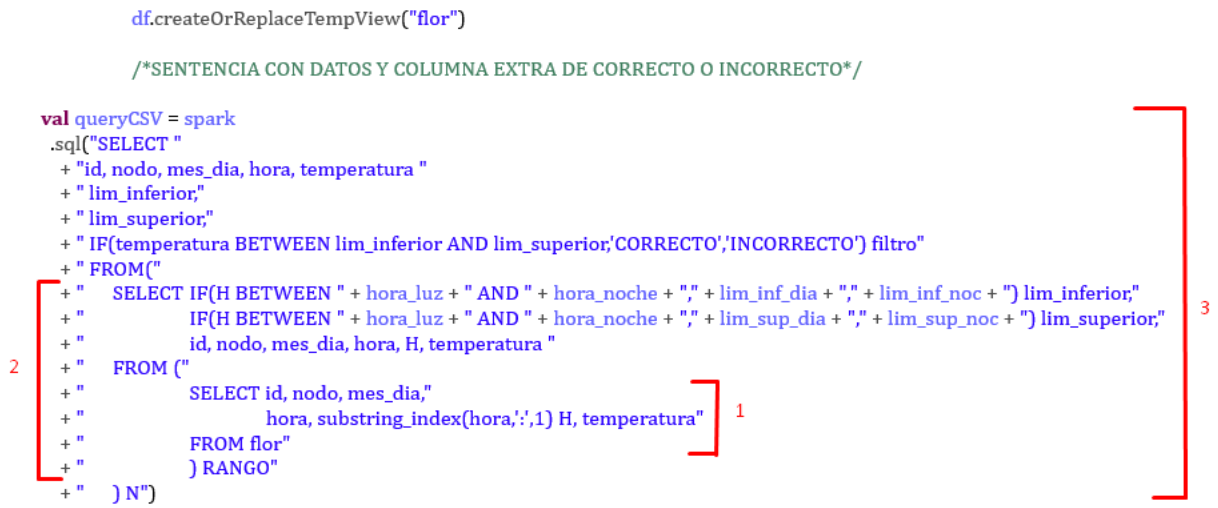


Figura 13. Aplicación de SQL sobre el batch de datos

A continuación, se brindará una breve explicación del modo en el que se están procesando los datos dentro de la sentencia de la Figura 13:

1. En el primer query que se realiza simplemente se toman todos los datos necesarios para la escritura de los datos de salida y se despeja la hora con la que ingresan los datos para comparar después según el requerimiento.
2. En este query, además de tomar los datos del query 1, se define cuál es el límite superior y límite inferior de la temperatura de acuerdo a la hora.
3. Finalmente, se hace una comparación del límite superior e inferior con respecto a la temperatura ingresada en los datos, y se escribe una nueva columna con la palabra CORRECTO si la temperatura se encuentra dentro de los límites e INCORRECTO si la temperatura se encuentra fuera de los límites.

Por último, los datos ya procesados deben almacenar en otros archivos con todos los datos que se pidió como resultado. A continuación, en la Figura 14 se mostrará la porción de código con

la que se envía a escribir archivos en el mismo formato .csv, con los resultados de los datos ya procesados.

```
queryCSV
 .writeStream
 .format("csv")
 .option("path", "/home/dalia/curso2/InvernaderoSpark/result/")
 .start()
 .awaitTermination()
```

Figura 14. Escritura de los archivos resultantes

Como se observa en el código de la Figura 14, además de la dirección y el formato con el que se quiere escribir los resultados, es en este instante donde se inserta la opción de comenzar con el análisis del stream y, a su vez, a esperar que el mismo termine.

4.4 RESULTADOS

Se puso a prueba el código explicado anteriormente, para lo cual se utilizaron datos reales obtenidos del invernadero. Para efectos de poner a prueba la herramienta y el código, los datos se fragmentaron en distintos archivos, que se agregarán paulatinamente y simplemente se realizarán pruebas de funcionamiento. Se ingresará un documento con los siguientes datos:

ID	Nodo	fecha	hora	temp	humedad	brillo	fact_uv	cal_aire
427	1	4/3/2018	17:37:41	14.3	72.3	74	1	39
428	1	4/3/2018	17:47:42	14.8	73.1	64	2	39
429	1	4/3/2018	17:57:42	15.3	70.9	71	1	38
430	1	4/3/2018	18:07:43	14.1	74.7	107	1	38
431	1	4/3/2018	18:17:43	13.6	77.9	133	1	39
432	1	4/3/2018	18:27:44	13.5	75.6	240	1	39
433	1	4/3/2018	18:37:45	13.3	76.5	522	1	38
434	1	4/3/2018	18:47:45	12.8	79.4	883	1	39
435	1	4/3/2018	18:57:46	12.7	80.2	954	1	39
436	1	4/3/2018	19:07:47	11.9	83.7	960	1	39
437	1	4/3/2018	19:17:47	12.6	82.2	962	1	39

Figura 15. Datos para pruebas de la solución

Al correr el código de Spark dentro del IDE de Eclipse, éste se queda esperando la terminación del mismo, mientras tanto siempre sigue escuchando por la aparición de otro resultado y siguen mostrando las lecturas dadas por cada procesamiento (aunque éste no se realice) cada vez, como se muestra en la Figura 16.

Figura 16. Log de stream escrito por Spark cada vez que realiza un proceso de batch

Una vez procesados los datos, el archivo guarda la información de la Figura 15, según el ejemplo propuesto. De esta manera, podemos verificar que si se compara la temperatura real con

respecto a: la hora del día deseada y con respecto a los límites solicitados el resultado se muestra claramente en la Figura 17.

ID	Nodo	fecha	hora	temp	lim_inf	lim_sup	filtro
427	1	4/3/2018	17:37:41	14.3	20	25	INCORRECTO
428	1	4/3/2018	17:47:42	14.8	20	25	INCORRECTO
429	1	4/3/2018	17:57:42	15.3	20	25	INCORRECTO
430	1	4/3/2018	18:07:43	14.1	20	25	INCORRECTO
431	1	4/3/2018	18:17:43	13.6	20	25	INCORRECTO
432	1	4/3/2018	18:27:44	13.5	20	25	INCORRECTO
433	1	4/3/2018	18:37:45	13.3	20	25	INCORRECTO
434	1	4/3/2018	18:47:45	12.8	20	25	INCORRECTO
435	1	4/3/2018	18:57:46	12.7	20	25	INCORRECTO
436	1	4/3/2018	19:07:47	11.9	10	12	CORRECTO
437	1	4/3/2018	19:17:47	12.6	10	12	INCORRECTO

Figura 17. Resultado de los datos procesados (títulos sobrepuestos)

Como se observa en la figura 17, los datos han sido procesados y validados como CORRECTOS e INCORRECTOS correspondientemente. Como parte de los requerimientos, se ha deseado incluir los parámetros de límite superior e inferior de temperatura, así el experto en el tema será capaz de visualizar más claramente el procesamiento.

Al procesar de una sola vez los 4278 datos se obtuvieron los siguientes resultados:

- **Cantidad de datos procesados por segundo:** 605,8032554847841 (606 aproximadamente) datos
    - **Tiempo total de procesamiento aproximado:** 13,0617 segundos
- **Resultado total de datos analizados:**

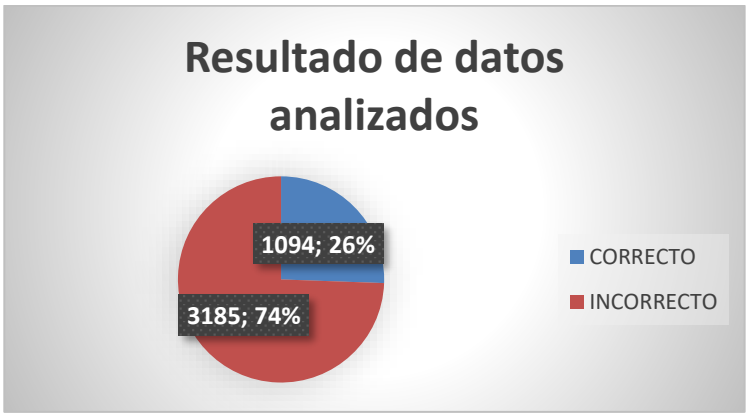


Figura 18. Resultado de la muestra total de datos analizados

Según se puede verificar en la figura 18, alrededor del 26% se consideran datos "Correctos" y el 74% datos "incorrectos". Esto quiere que, de la muestra asignada para la verificación, el porcentaje de "correctos" pareciera ser bajo, pero en realidad ese porcentaje refleja la temperatura ideal, mas no la única válida para efectos reales.

- **Resultado de datos en horas de luz:**

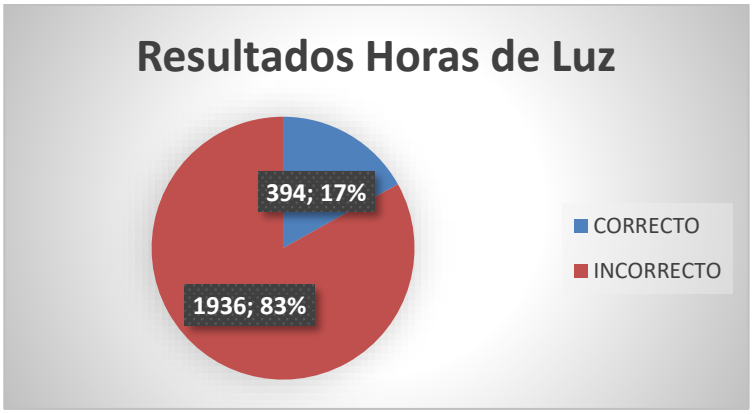


Figura 19. Resultado horas de luz

- **Resultados de datos en horas de oscuridad:**

Figura 20. Resultados horas de oscuridad

- **Resultados de datos horas de luz vs. horas de oscuridad:**

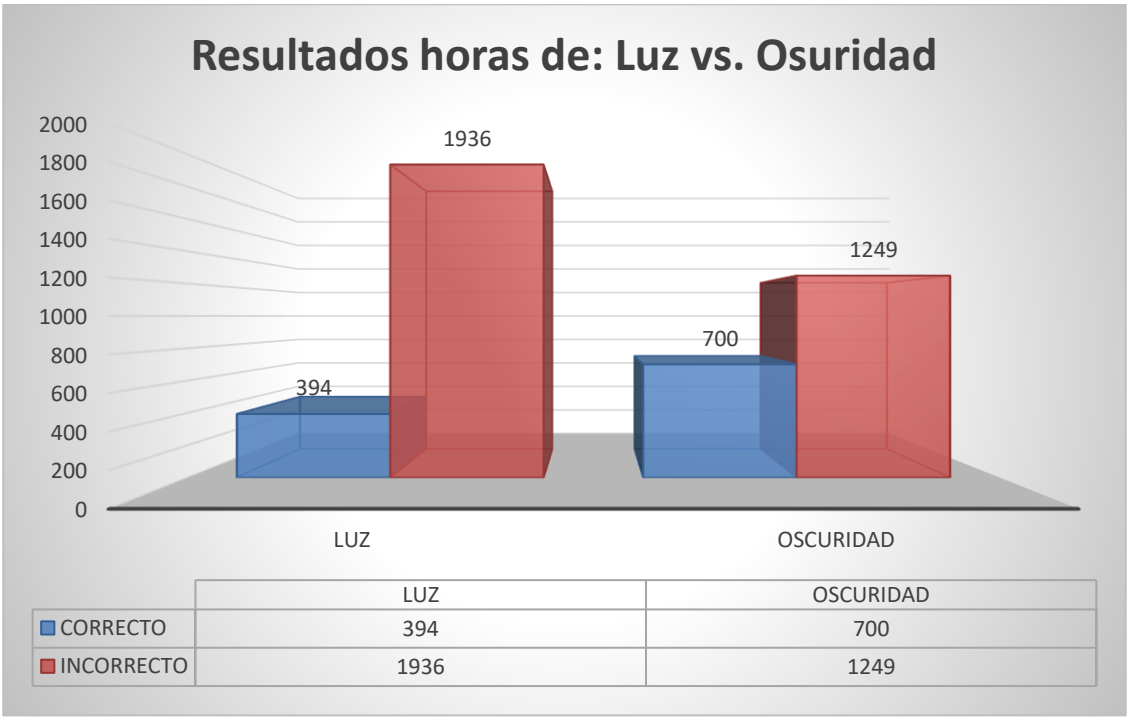


Figura 21. Resultados: horas de luz vs. horas oscuridad

Si se analiza las diferencias de la figura 21 tenemos que, los datos tomados en las horas de oscuridad como “correctos” o “incorrectos”, son superiores a los datos tomados en las horas de luz.

Esto porque lo tomado en horas de oscuridad representa el 63.99% del total de datos “correctos” versus el 36.01% de datos en horas de luz; además que el número de datos “incorrectos” en horas de oscuridad (39.22% del total de la muestra) es menor al número de datos “incorrectos“ en horas de claridad (60.78% del total de la muestra).

CAPÍTULO V
CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Según el análisis realizado bajo la norma ISO 25000, entre las herramientas Spark Streaming y Apache Flink, se llegó a la conclusión que Spark Streaming es más consistente en cuanto a comportamiento y tiempos de procesamiento según los datos obtenidos en el invernadero de rosas del IASA I, a pesar de que Apache Flink demostró mejor resultado en el uso de recursos durante el procesamiento.
- La aplicación de la herramienta mejor puntuada en el análisis comparativo, Spark streaming, para búsquedas efectivas en datastream de invernaderos fue un éxito, puesto que permite llegar a la solución concreta solicitada por los expertos en el campo de crecimiento de rosas, considerando el tipo de flujo de datos y el modelo con el que se trabaja actualmente.
- No se descarta que puedan existir ajustes a futuro que puntualicen de mejor manera los requerimientos individuales del invernadero en el que se esté trabajando (ajustes más precisos en horas, por ejemplo).
- La solución implementada con los datos de los sensores del invernadero es consistente, lógica y útil para proyectos a futuro; así como también permitió el filtrado de los datos de manera rápida y correcta.
- De la muestra tomada para analizar los datos se encontró que el 26% eran datos considerados “correctos” y el 74% eran datos considerados “incorrectos”. Sin embargo, no debe tomarse dichos datos como absolutos, puesto que únicamente representan el ideal en el crecimiento de las rosas.

5.2 RECOMENDACIONES

- Se recomienda poner a prueba la herramienta utilizando otros datos, es decir, implementar otra solución con respecto a un sensor/dato diferente a temperatura para poner a prueba con diferentes tipos de procesamiento a la herramienta.

- Se recomienda realizar más algoritmos o scripts que difiera en el procesamiento propuesto en esta solución, esto para tomar experiencia con la herramienta y poder explotarla al máximo.
- Se recomienda realizar soluciones con versiones mejoradas o superiores a la utilizada en el presente proyecto, puesto que Spark brinda soporte y ha introducido innovaciones que lo hacen aún más competitivo con sus similares.

REFERENCIAS BIBLIOGRAFICAS

Apache Flink. (2016). *Apache Flink*. Obtenido de Apache Flink: http://flink.apache.org/index.html

Apache Kafka. (2017). *Apache Kafka*. Obtenido de Apache Kafka: https://kafka.apache.org/intro

Apache Spark. (Mayo de 2017). *Apache Spark*. Obtenido de Apache Spark: http://spark.apache.org/docs/latest/index.html

Asociación de Productores y Exportadores de Flores. (21 de Mayo de 2017). *Expoflores*. Obtenido de Clima bajo invernadero: http://flor.ebizor.com/clima-bajo-invernadero/

Babcock, B., Babu, S., Datar, M., Motwani, R., & Widom, J. (2002). Models and Issues in Dat Stream Systems. *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Data Systems (PODS '02)*.

BBVA. (07 de agosto de 2015). *BBVAOPEN4U*. Obtenido de Tres alternativas sólidas de Big Data en tiempo real: Spark, Storm y DataTorrent: https://bbvaopen4u.com/es/actualidad/tres-alternativas-solidas-de-big-data-en-tiempo-real-spark-storm-y-datatorrent-rts

Boyd, D., & Crawford, K. (2012). Critical Questions For Big Data. *Taylor & Fancis Online*.

Callau Zori, M. (2012). *Algoritmos de agrupación para flujos de datos en entornos centralizados y distribuidos.* Obtenido de Universidad Politécnica de Madrid: http://oa.upm.es/22467/1/MAR_CALLAU_ZORI.pdf

Diván, M. (15 de Julio de 2011). *Enfoque Integrado de Procesamiento de Flujos de Datos Centrado en Metadatos de Mediciones*. Obtenido de Postgrado Facultad de Informática: http://postgrado.info.unlp.edu.ar/Carreras/Doctorado/Tesis/Divan_Mario_Jose.pdf

École Polytechnique Fédérale. (2017). *Scala*. Obtenido de Scala: http://docs.scala-lang.org/tour/tour-of-scala.html

El Telégrafo. (2 de Noviembre de 2017). *El Telégrafo*. Obtenido de El Telégrafo: http://www.eltelegrafo.com.ec/noticias/economia/8/el-sector-floricola-no-se-recupera-desde-2014

International Standarization Organization ISO 25000. (2017). Norma ISO/IEC 25000. *Portal ISO 25000*.

International Standarization Organization ISO 9126. (1991). Norma ISO/IEC 9126. *Norma ISO/IEC 9126*.

La hora. (20 de Julio de 2003). Florícolas en quiebra. *La Hora*.

La hora. (31 de Enero de 2016). Florícolas de Ecuador en crisis. *La Hora*.

Madden, S. (2012). From Databases to Big Data. *IEEE Xplore*.

Mohammad, W. (2013). Big data: Issues, challenges, tools and Good practices. *IEEE Xplore*.

Perdigones, A., Peralta, I., Nolasco, J., Muños, M., & Pascual, V. (2004). Sensores para el control climático en Invernadero. *Horticom.com*.

Pérez Esteso , M. (2015). *Universidad Politécnica de Madrid* . Obtenido de Análisis de Arquitecturas de Procesado de Streaming Big Data: http://www.dit.upm.es/~posgrado/doc/TFM/TFMs2014-2015/TFM_Mario_Perez_Esteso_2015.pdf

Portabales Goberna, A., & Martínez Álvarez, R. (02 de 11 de 2017). *Big Data aplicado: Spark vs Flink*. Obtenido de Gradiant: https://www.gradiant.org/blog/big-data-aplicado-spark-vs-flink-2/

Postigo-Boix, M., Aguilar-Igartua, M., & García-Haro, J. (2014). Transmisión eficiente de Bloques en Tiempo Real sobre Redes IP. *Research Gate.* Obtenido de https://www.researchgate.net/profile/Marcos_Postigo-Boix/publication/257329710_Transmision_Eficiente_de_Bloques_en_Tiempo_Real_sobre_Redes_IP/links/5410304a0cf2df04e75b6ff1.pdf

Russom, P. (2011). Big Data Analytics. *TDWI Best Practices Report*.