ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"Utilización de la plataforma Hadoop para la implementación de un programa que permita determinar mensajes spam"

INFORME DE MATERIA DE GRADUACIÓN

Previo a la obtención del Título de:

INGENIERO EN CIENCIAS COMPUTACIONALES

ESPECIALIZACIÓN EN SISTEMAS DE INFORMACION

Autores:

GUSTAVO JAVIER CRESPO PINANCELA

SUSANA MARIA VELIZ MONCADA

GUAYAQUIL – ECUADOR

Año: 2012

DEDICATORIA

A Dios por darme la oportunidad de vivir esta vida y por el amor que nos brinda cada día, a mi padre, el hombre trabajador de quien siempre me he inspirado, por sus ganas y su lucha, a mi madre, luchadora implacable, que me lo ha dado todo, fuerzas, ánimos, esperanzas, y que ha hecho de nosotros lo que ahora somos, a mi hermana que fue mi ejemplo a seguir por su rectitud y por su altruista ser, a mi hermano del cual yo he aprendido muchas cosas y al cual espero haberle enseñado unas cuantas también, y por último a todos mis profesores que en la vida estudiantil me han guiado y han dejado una huella en mi persona. Para todos ustedes.

GUSTAVO CRESPO P.

A Dios por iluminar mi camino a cada instante, por darme la sabiduría necesaria para afrontar las adversidades, a mi padre por apoyarme siempre en mis estudios mientras estuvo con vida, a mi madre por tantas noches de desvelo a mi lado impulsándome a continuar dándome siempre una luz de esperanza con sus palabras de ánimo con su apoyo y cariño, a mis dos hermanas que fueron mi ejemplo a seguir por su apoyo incondicional tanto económico como emocional, a mis jefes por darme las facilidades de seguir estudiando, a mis mejores amigos, mis primos y todas aquellas personas que colaboraron para que culmine con éxito tan maravilloso sueño.

SUSANA VELIZ M.

AGRADECIMIENTO

A todas aquellas personas que de una u otra forma nos sirvieron de guía para la culminación del presente trabajo pero en especial ofrecemos nuestro más sincero agradecimiento a la Ingeniera Vanessa Cedeño Mieles y el Ingeniero Javier Tibau, por su constante e invaluable colaboración.

DECLARATORIA EXPRESA

"La responsabilidad del contenido de este Trabajo de Graduación, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la **Escuela Superior Politécnica del Litoral**"

(Reglamento de Graduación de la ESPOL)

GUSTAVO CRESPO PINANCELA.

SUSANA VELIZ MONCADA.

TRIBUNAL DE SUSTENTACIÓN

Ing. Vanessa Cedeño.

PROFESORA DE LA MATERIA DE GRADUACION

Ing. Javier Tibau.

PROFESOR DELEGADO POR EL DECANO

RESUMEN

Se denomina spam a los mensajes no solicitados, de remitente no conocido que perjudican de alguna o varias maneras al receptor. Habitualmente son de tipo publicitario y enviados en cantidades masivas.

Este proyecto propone un modelo de análisis de los archivos que sigan el formato de un correo electrónico, según la norma RFC822 con el fin de determinar posibles mensajes spam. Para el filtrado de los mismos hemos utilizado la plataforma Apache Hadoop junto con la plataforma para analizar grandes cantidades de datos en un lenguaje de alto nivel Apache Pig.

Para este estudio, obtuvimos una cantidad predeterminada de correos con y sin spam. Se realizó un análisis utilizando la metodología de filtros bayesianos aplicados a los mensajes electrónicas introducida por el ensayista, programador, diseñador de lenguajes y co-fundadador de viaweb Paul Graham, en modo stand alone así como en multinodo para ver las diferencias de tiempos de ejecución con uno y varios computadores. El conocimiento previo de la cantidad exacta de spam nos permitió determinar el grado de exactitud de nuestro filtro.

INDICE GENERAL

RESUMEN VII

INDICE GENERAL VIII

INDICE DE FIGURAS XII

INDICE DE TABLAS XIII

ABREVIATURAS XIV

INTRODUCCIÓN XV

CAPÍTULO 1 1

1. ANTECEDENTES Y JUSTIFICACIÓN. 1

1.1. CONCEPTO E HISTORIA DEL SPAM 1

EVOLUCIÓN DEL SPAM A TRAVÉS DEL TIEMPO. 2

ANÁLISIS DE CONTENIDOS 2

TEXTO SIMPLE Y HTML 3

MENSAJES PERSONALIZADOS 3

RENGLONES DE TEXTO ALEATORIO Y TEXTO INVISIBLE 4

GRÁFICOS 4

TEXTOS PARAFRASEADOS 5

1.2. JUSTIFICACIÓN 5

ABUSO DE LOS RECURSOS DE OTROS 5

FRAUDE 6

ÉTICA 7

1.3. OBJETIVOS 8

1.3.1. Clúster utilizando Apache Hadoop. 8

1.3.2. Algoritmos para detección de mensajes spam utilizando filtro bayesiano y probabilidades. 9

1.3.3. Reconocimiento de mensajes Spam. 10

1.3.4. Aplicación utilizando Hadoop, Pig y probabilidades del filtro bayesiano para detectar mensajes spam. 11

1.3.5. Arquitectura y descripción del manejo de los datos en las funciones MapReduce – Apache Pig. 15

1.4. ALCANCE 16

Delimitación del proyecto para cumplir los objetivos. 16

CAPÍTULO 2 17

2. FUNDAMENTOS TEÓRICOS DE HADOOP. 17

2.1. APACHE HADOOP. 17

2.1.1. Componentes de Hadoop. 18

2.1.2. Definición de Distributed File System (HDFS). 19

2.1.3. Definición de MapReduce. 22

2.2. APACHE PIG. 24

CAPÍTULO 3 26

3. ANÁLISIS DE LA SOLUCIÓN. 26

3.1. REQUERIMIENTOS. 27

3.1.1. Requerimientos Hardware. 27

3.1.2. Requerimientos Software. 28

3.2. ANÁLISIS DE LAS CAPACIDADES DEL FRAMEWORK A UTILIZAR..... 28

3.3. REQUERIMIENTOS Y DATOS O MENSAJES CORREOS GENERADOS O ALMACENADOS. 30

CAPÍTULO 4 31

4. DISEÑO E IMPLEMENTACIÓN DE HADOOP PARA COMBATIR EL SPAM. 31

4.1. MODELO LÓGICO DEL PROCESO DE IDENTIFICACIÓN DE SPAM. 31

4.2. MODELAMIENTO DE LOS DATOS PARA SU PROCESAMIENTO. 33

4.3. PLAN DE PRUEBAS. 34

4.4. IMPLEMENTACIÓN DE APACHE HADOOP, APACHE PIG Y MAPREDUCE PARA COMBATIR EL SPAM. 35

4.4.1. Herramientas a Utilizarse. 35

4.4.2. Instalación y configuración de las herramientas a utilizarse. ... 36

CAPÍTULO 5 43

5. PRUEBAS Y RESULTADOS 43

5.1. EJECUCIÓN DE LAS PRUEBAS 43

(Ver los detalles de las pruebas en los anexos) 43

Prueba 1 44

Prueba 2 45

Prueba 3 46

5.2. ANÁLISIS DE LOS RESULTADOS 47

RECOMENDACIONES 55

ANEXOS 57

ANEXO A 58

ANEXO B 59

INDICE DE FIGURAS

Figura 1-1 – Clúster multi-nodo 10

Figura 1-2 – Ej. Administración y planeamiento común del flujo de datos Apache Pig 15

Figura 2-3 – Logo Hadoop 17

Figura 2-4 – Arquitectura HDFS 21

Figura 2-5 – Logo de Pig 24

Figura 3-6 - Esquema de la solución 26

Figura 5-7 – Comparación de la Duración contra la cantidad de nodos 48

Figura 5-8 – Representación de la ecuación de regresión lineal. 49

Figura 5-9 – Cantidad de correos detectados | Corpus real – Prueba 50

Figura 5-10 – Porcentaje de eficacia filtro | Falsos negativos – Prueba 50

Figura 5-11 – Porcentaje de eficacia filtro |Falsos positivos – Prueba 51

INDICE DE TABLAS

Tabla I – Resultados prueba # 1 44

Tabla II – Eficiencia Filtro| Falsos Negativos # 1 44

Tabla III – Eficiencia Filtro| Falsos Positivos # 1 44

Tabla IV – Resultados prueba # 2 45

Tabla V – Eficiencia Filtro| Falsos Negativos # 2 45

Tabla VI – Eficiencia Filtro| Falsos Positivos # 2 45

Tabla VII – Resultados prueba # 3 46

Tabla VIII – Eficiencia Filtro| Falsos Negativos # 3 46

Tabla IX – Eficiencia Filtro| Falsos Positivos # 3 46

ABREVIATURAS

RBL	Real-Time Black List
GB	Gigabyte
HTML	HyperText Markup Language
ISP	Internet Service Provider
HDFS	Hadoop Distributed File System
ESPOL	Escuela Superior Politécnica del Literal
XML	Extensible Markup Language

INTRODUCCIÓN

Anti-Spam utilizando Hadoop.

En el transcurso de los años de la última década las cuentas de correo electrónico han sido afectadas por Spam de correo no solicitado, anónimo y masivo, los cuales han sido combatidos con diferentes filtros de spam, dichos spam usan direcciones de correo falsas o que pertenecen a otros, los spammers ganan dinero con el pequeño porcentaje que les responde, por eso la mayoría de las veces envían correos a grandes escales.

Las técnicas de los spammers evolucionan cada vez que un filtro de correo electrónico es mejorado, ya que cada vez que una compañía de seguridad informática desarrolla un filtro con mayor efectividad en la detección de los mismos, los spammers reinvierten sus ganancias para implementar nuevas técnicas que les permitan evadir estos filtros, lo cual lo convierte un ciclo interminable de tácticas entre los spammers y las empresas.

La detección de posibles correos spam es un problema para el cual diseñamos un filtro bayesiano basado en probabilidades e Inteligencia Artificial, en el cual se realiza un entrenamiento previo a el filtro para que pueda aprender y detectar a lo que nosotros denominamos spam. Y luego

con estos datos y probabilidades realizar un reconocimiento de cada mensaje electrónico y etiquetar con una probabilidad a cada uno de ellos.

El contenido de este trabajo se distribuye de la siguiente manera: En el capítulo 1 se describe los antecedentes (concepto e historia del spam), justificación (análisis del problema), objetivos y alcance del presente trabajo. En el capítulo 2 se presentan los fundamentos teóricos de HADOOP. En el capítulo 3 se describe el análisis de la solución. En el capítulo 4 se muestra el diseño y se detalla la implementación de hadoop para combatir el spam. Finalmente, las pruebas y resultados son mostrados en el capítulo 5.

CAPÍTULO 1

1. ANTECEDENTES Y JUSTIFICACIÓN.

1.1. Concepto e historia del Spam

El Spam es correo no solicitado, anónimo y masivo. [1]

Correo Anónimo

El spam es enviado a través de direcciones de remitentes falsas o pertenecientes a otras personas con el fin de ocultar la identidad del verdadero remitente.

Envíos masivos

El spam es enviado en cantidades masivas debido a que los spammers hacen dinero con el pequeño porcentaje de destinatarios que responden. Por eso, para ser efectivo, los envíos iníciales tienen que ser de gran volumen.

No solicitado

Un mismo mensaje de correo puede ser clasificado como spam o como correspondencia legítima dependiendo de si el usuario ha escogido recibirlo o no, tales como listas de correos, noticias y otros materiales de publicidad.

Podemos definir el spam como publicidad masiva no solicitada que es enviada y recibida a través del correo electrónico, la cual hizo una de sus primeras apariciones a mediados de la época de los noventa, prácticamente sucedió inmediatamente después de que la suficiente cantidad de personas empezó a usar el correo electrónico.

Alrededor del año 1997, recién se empezó a considerar que el spam era un problema, y en este mismo año apareció la primera RBL. Las técnicas de anti-detección de los spammers han ido evolucionado constantemente en respuesta a la aparición de cada filtro mejorado. Y es así que tan rápido como las empresas de seguridad diseñan mejores y efectivos filtros, los spammers también cambian sus técnicas e inventan nuevas maneras para evadirlos. Lo cual ha creado a un círculo vicioso, en el que los spammers reinvierten sus ganancias en el diseño de nuevas formas para eludir los filtros de spam. [1]

Evolución del Spam a través del tiempo.

Análisis de contenidos

En la actualidad la mayoría de filtros y técnicas anti spam son basados en el análisis de contenido de los mensajes, asi como lo representan el encabezado, el texto y lo archivos adjuntos.

Para lo cual las personas que se dedican a la actividad de generar y enviar dichos mensajes, son conscientes de esta variedad de filtros para lo cual desarrollan contenidos que puedan evadir dichos filtros.

Texto simple y HTML

Los primeros spam eran muy simples, consistían en mensajes idénticos de correo electrónico que eran enviados a todos los destinatarios de una determinada lista de correo. Estos mensajes eran demasiado fáciles de filtrar, gracias a la igualdad de su texto.

Mensajes personalizados

Con el tiempo, los spammers fueron incluyeron en los mensajes una forma de saludo que se basada en la dirección de correo del destinatario. Como cada uno de los mensajes contenía un saludo personalizado, los filtros que identificaban mensajes idénticos no detectaban este nuevo tipo de spam. [1]

Fue entonces que los expertos en seguridad empezaron a diseñar filtros que pudieran identificar renglones idénticos, los cuales si eran nuevos podrían ser agregados a las reglas de filtración ya establecidas. También a través de técnicas de lógica difusa, detectaban textos ligeramente modificados. [1]

Renglones de texto aleatorio y texto invisible

En la actualidad, para eludir los filtros de spam, los spammers incluyen renglones de texto muy parecidos a la de la correspondencia de negocios legítima o renglones de texto generado aleatoriamente al inicio o al fin de los correos. Otra forma de eludir los filtros de spam es incluyendo texto prácticamente invisible en los mensajes en formato HTML en dos métodos: con texto muy pequeño, o utilizando el mismo color de fondo en las letras. [1]

Cualquiera de los dos métodos evade con éxito los filtros de contenido y estadísticos. Lo cual impulsó el desarrollo de motores de búsqueda, que realizaban un scan de los mensajes buscando estos tipos de textos, y que a su vez analizaban detalladamente el HTML y el contenido de los mismos. En la mayoría de las soluciones anti spam se podían detectar este tipo de trucos sin necesidad de analizar todo el contenido de cada uno de mensajes de correo. [1]

Gráficos

Este tipo de spam es muy complicado de detectar. En la actualidad se están diseñando métodos que puedan extraer y analizar el texto embebido en este tipo de archivos. [1]

Textos parafraseados

Un mismo anuncio puede ser parafraseado una infinidad de veces, haciendo que cada uno de estos mensajes aparente ser uno totalmente legítimo. Como era de esperarse, los filtros anti spam deben de recopilar una gran cantidad de muestras antes de estos sean considerados como spam. [1]

1.2. Justificación

Abuso de los recursos de otros

Cuando un spammer realiza el envío de un mensaje a un gran número de personas, este es llevado por una diversidad de sistemas en el camino hasta su destinatario, transfiriendo el costo a alguien totalmente diferente de quien en realidad lo origina. [2]

Quienes por desgracia están involucrados de alguna forma con el medio de transmisión repentinamente asumen el costo de transferir la publicidad que envía el spammer. [2]

El número de correos spam enviado cada día es increíblemente exorbitante. No existe ninguna justificación para que inocentes asuman el costo de estos mensajes no solicitados. [2]

Los métodos que utilizan los spammers para eludir la responsabilidad de sus acciones son en la mayoría de las veces fraudulentos y tortuosos. Una gran cantidad de juicios están en curso entre los spammers y las víctimas inocentes que fueron sometidas a este tipo de inundaciones. [2]

Fraude

Los spammers están conscientes de que la mayoría de los que reciben sus correos por lo general no les interesa en lo absoluto el contenido de los mismos. Por lo cual, para intentar que incrementen sus "llegadas", recurren a todo tipo de artimañas para hacer que las personas abran sus correos. [2]

Por ejemplo, arman el asunto del correo de tal manera que parezca de algún conocido, o una de alguna de sus respuestas. Tratan de disfrazarlo de tal forma que no se pueda descartar como una publicidad. [2]

En la mayor parte de los casos los ISPs y sus Clientes desarrollan filtros que faciliten el manejo del correo. Mientras los filtros generalmente consumen los recursos en el ISP, hacen más lento el envío y entrega del correo, así como la navegación, ayudan a los

Clientes a utilizar lo mejor posible su cuenta de correo. Los spammers conocen esto así que tratan de ocultar lo más que pueden el verdadero origen del mensaje. [2]

Una de las tácticas más utilizadas es hacer que la transmisión y la entrega la haga el servidor de un tercer inocente. Esta táctica incrementa los daños al doble: ambas partes, el relay y el receptor son inundados con correo spam. En donde las quejas por los correos spam recibidos las recibe el tercer inocente, porque aparentemente este los origina. [2]

Ética

El spam se fundamenta en: robo de servicios, fraude y engaño, mediante la transferencia del costo de quien lo envía (el spammer) a quien lo recibe (la víctima). Aunque la mayor parte de los productos y servicios que se ofrecen no fuesen de dudosa legalidad, un negocio que toma algo de sus potenciales Clientes sin su autorización previa, que se aprovecha de los incautos e inocentes, y abusan de los recursos del Internet estuvo, está y estará, condenado al más rotundo de los fracasos. [2]

1.3. Objetivos

- Diseñar un modelo de análisis de los archivos que sigan el formato de un correo electrónico, según la norma RFC822.
- Realizar un filtro utilizando Apache Hadoop como plataforma principal utilizando sus características de computación, almacenamiento paralelo, junto con Apache Pig el cual nos proporciona un lenguaje de alto nivel llamado Pig Latin con el cual es realizado el filtro bayesiano propuesto, utilizando métodos de probabilidades para determinar luego del entrenamiento, la probabilidad de que un mensaje sea un spam.
- Analizar la eficacia del filtro, basándonos en la cantidad de mensajes spam y mensajes no spam que detecto contrastándolos con la cantidad real de mensajes spam y no spam.

1.3.1. Clúster utilizando Apache Hadoop.

El clúster usando hadoop incluye un “single master”, y múltiples “worker nodes”. El nodo maestro corre los demonios de hadoop JobTracker, y Namenode.

Un esclavo o nodo trabajador actúa como un nodo de datos y un rastreador de tareas, en el cual correrán los demonios Datanode y Tasktracker.

El nodo secundario, representa el nodo backup del nodo maestro, debido a que en este se almacena la información del nodo maestro, si es que este llega a fallar. Sobre este nodo corre el demonio Secondary NameNode.

En grandes clúster, el HDFS es administrado a través de un servidor NameNode dedicado para alojar los índices de sistemas de archivos y un secundario NameNode que puede generar “snapshots” de las estructuras de memoria del NameNode, para prevenir corrupción en los archivos de sistema y reducir perdida de datos. Similarmente un servidor “JobTracker” independiente puede administrar la planificación de tareas. [3]

1.3.2. Algoritmos para detección de mensajes spam utilizando filtro bayesiano y probabilidades.

Para la detección de mensajes spam, diseñamos un filtro bayesiano, basado en las características del filtro propuesto por Paul Graham en el ensayo “A Plan for Spam”. [4]

El cual básicamente es creado utilizando un corpus de mensajes spam y mensajes no spam, en el cual se analiza la probabilidad de que una palabra contenida en un mensaje, represente que este mensaje sea o no un spam, con una cierta probabilidad.

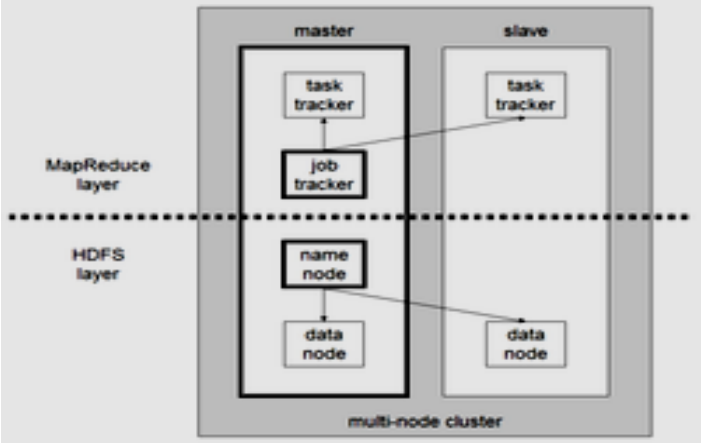


Figura 1-1 – Clúster multi-nodo. [3]

1.3.3. Reconocimiento de mensajes Spam.

Un mensaje será reconocido para nuestro filtro como spam si la probabilidad de spam obtenida por el filtro, según las palabras que contenga devuelva como resultado un índice mayor o igual al 90%.

Opción anterior

Técnicas de SQL, debido a que los correos correo, se almacenaran en nuestro repositorio de HIVE, como registros en bases de datos, lo cual nos permite tener acceso, manipulación y administración a través de sentencias SQL.

Opción actual

Filtro bayesiano implementado sobre el lenguaje de alto nivel Apache Pig, que opera sobre Apache Hadoop para el almacenamiento y computación paralela, así como Apache Tika, como parser principal para los datos de entrada, que siguen un formato RFC822.

1.3.4. Aplicación utilizando Hadoop, Pig y probabilidades del filtro bayesiano para detectar mensajes spam.

La aplicación final, como resultado de las investigaciones, análisis y desarrollo de técnicas y uso de herramientas para la identificación de spam, consta de varios demonios detallados a continuación:

El demonio maestro llamado NameNode, el más vital demonio de Hadoop. Debido a que Hadoop emplea una arquitectura maestro-esclavo tanto para el almacenamiento distribuido así como para la computación distribuida, para lo cual al almacenamiento distribuido conocido como Hadoop Distributed File System o HDFS, es entonces cuando entra en acción el NameNode el cual es el maestro del HDFS que es el encargado de administrar la metadata, así como la administración de las direcciones de bloques en donde reside los datos dentro del HDFS.

Básicamente el NameNode es el bookkeeper del HDFS, este mantiene almacenado la pista o rastro de cómo los archivos fueron divididos en bloques de archivos más pequeños así como cuáles nodos almacenan estos bloques y sobre todo la “vitalidad” del sistemas de archivos distribuido. [5]

La función del NameNode entonces es memoria y uso intensivo de entrada y salida. Cabe recalcar que el host que va servir como alojamiento para el NameNode, no es recomendable utilizarlo para albergar algún otro demonio como el DataNode o TaskTracker.

Tal como hemos podido revisar la importancia de este demonio, es exactamente en este punto donde radica un aspecto negativo por así decirlo, debido a que este es el único punto de fallo para el Clúster de Hadoop, porque ya sea que cualquiera de los otros demonios fallen, ya sea por razones de fallas de software o hardware, el clúster de Hadoop podrá continuar sin problemas o si se desea se puede reiniciar, pero esto no pasa para cuando el NameNode falla. La solución a este problema se explicara a continuación en la descripción del Secondary NameNode.

El Secondary NameNode, es un demonio asistente para monitorear el estado del clúster HDFS. Así como el NameNode, cada clúster tiene un Secondary NameNode y típicamente reside en su propia máquina o por lo menos esto es lo que se recomienda.

El Secondary NameNode difiere al NameNode, en que este no procesa ni recibe ningún cambio o registro en tiempo real del HDFS. En vez de esto se comunica con el NameNode para tomar snapshots de la meta data, en intervalos de tiempo definidos en la configuración del clúster y es así como el Secondary NameNode ayuda a minimizar el tiempo de caída y la pérdida de datos en caso de que el NameNode falle.

El DataNode es el demonio que cada nodo esclavo tendrá corriendo en su máquina para realizar el trabajo de campo de los sistemas de archivos distribuido.

Cuando se quiere leer o escribir un archivo HDFS, el archivo es dividido en bloques y el NameNode le dirá al cliente DataNode donde cada reside cada bloque.

El cliente se comunica directamente con el demonio DataNode para procesar los archivos locales correspondientes a los bloques. Además, un DataNode puede comunicarse con otros DataNode para replicar sus bloques de datos para redundancia.

Los demonios JobTracker y TaskTracker representan en Hadoop la computación distribuida, donde siguiendo la arquitectura maestro/esclavo, el JobTracker es el maestro de la supervisión de la ejecución total de los trabajos MapReduce, mientras que el TaskTracker administra la ejecución de tareas individuales sobre el nodo esclavo.

Una vez que se entrega el código a él clúster, el JobTracker determina el plan de ejecución determinando cual archivos procesar, asignando diferentes nodos para diferentes tareas y monitorizar todas las tareas que están corriendo. En caso que una tarea falle, el JobTracker automáticamente relanzara la tarea, posiblemente a otro diferente nodo, dependiendo de predefinidos reintentos.

El TaskTracker a su vez es el responsable de ejecutar las individuales tareas que el JobTracker asigna. Una de las más importantes responsabilidades del TaskTracker es constantemente comunicar con

el JobTracker. Así, si el JobTracker falla en la recepción de comunicación del TaskTracker dentro de un especificado tiempo, se asumirá que el TaskTracker ha fallado y se re enviara esta tarea a otro nodo en el clúster. [6]

1.3.5. Arquitectura y descripción del manejo de los datos en las funciones MapReduce – Apache Pig.

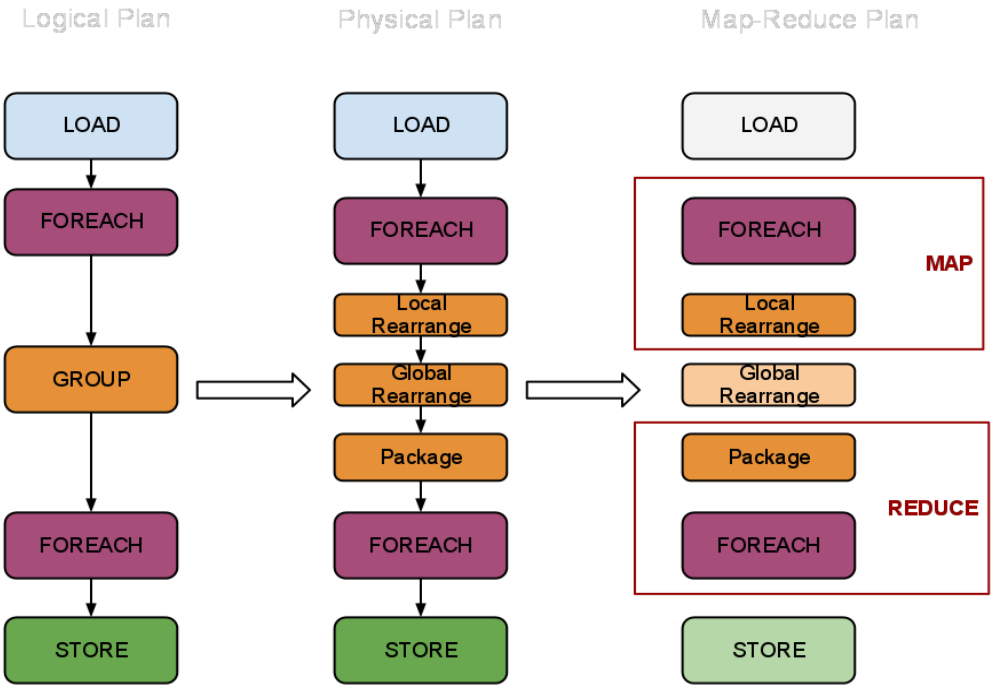


Figura 1-2 – Ej. Administración y planeamiento común del flujo de datos Apache Pig.[14]

Debido a el lenguaje de alto nivel de Pig Latin, en el cual es desarrollado nuestro filtro bayesiano, las operaciones de mapeo y

reducción son básicamente transparentes para nosotros, debido a que Apache Pig administra y maneja la creación de los mappers y reducers, según las ordenes indicadas en la secuencias de instrucciones que son convertidas directamente en jobs o trabajos de Hadoop automáticamente contenidas en el script.

Por lo cual podríamos decir que Apache Pig, resuelve el conflicto de paralelismo de mapeo y reducción de bajo nivel, dejándonos concentrar en la implementación de la solución, mas no las funciones de map y reduce propias de Hadoop

1.4. Alcance

Delimitación del proyecto para cumplir los objetivos.

Aplicación utilizando Apache Hadoop, Apache Pig capaz de estimar con un índice de probabilidad de spam a un mensaje de correo electrónico, utilizando un filtro bayesiano, creado y entrenado de antemano, que almacena un conjunto de palabras con probabilidades individuales.

La aplicación podrá analizar, mapear, reducir y estimar el índice de probabilidad de spam al final de la ejecución un listado con ids de los mensajes ingresados, así como su respectiva probabilidad.

CAPÍTULO 2

2. FUNDAMENTOS TEÓRICOS DE HADOOP.

2.1. Apache Hadoop.

Figura 2-3 – Logo Hadoop [4]

Software de código abierto para computación confiable, escalable y distribuida.

Como todo sistema de archivos distribuidos hadoop tiene como principal objetivo dar solución al problema de almacenamiento de información que sobrepase las capacidades de una máquina convencional.

Para la solución del mismo, gestionará y permitirá el almacenamiento de los datos en diferentes máquinas interconectadas a través de una red, haciendo que el proceso sea de total transparencia para el usuario sin que tenga que sumergirse en la complejidad de su proceso interno.

2.1.1. Componentes de Hadoop.

Sus librerías son un framework que permite el procesamiento distribuido de grandes conjuntos de datos a través de clúster de ordenadores mediante un simple modelo de programación. [10]

En lugar de confiar en el hardware para ofrecer mayor confiabilidad, la librería en sí está diseñada para detectar y controlar los errores en la capa de aplicación, para entregar un servicio de alta disponibilidad en la parte superior de un clúster de computadoras, cada una de las cuales están propensas a fallas. [4]

Incluye los siguientes componentes:

- **Hadoop Common:** Utilidades comunes que apoyan los otros componentes de Hadoop.
- **Hadoop Distributed File System (HDFS):** Sistema de archivos distribuidos que proporciona un alto rendimiento de acceso a datos de la aplicación.
- **Hadoop MapReduce:** Framework para el procesamiento distribuido de grandes conjuntos de datos en clúster de cómputo. [4]

2.1.2. Definición de Distributed File System (HDFS).

Es el sistema de almacenamiento primario utilizado por las aplicaciones Hadoop. HDFS crea varias réplicas de los bloques de datos y los distribuye a los nodos de cálculo a través de un clúster que permitan cálculos confiables y extremadamente rápidos. Un clúster HDFS consiste principalmente en un NameNode que gestiona el sistema de archivos de metadatos y DataNodes que almacenan los datos reales. HDFS es altamente tolerante a fallos y está diseñado para ser implementado en hardware de bajo costo. HDFS proporciona un acceso de alto rendimiento para los datos de aplicación y es adecuado para aplicaciones que tienen grandes conjuntos de datos. [5]

Un ejemplo HDFS puede consistir en cientos o miles de máquinas servidores, cada parte almacenada en un sistema de archivos de datos. El hecho de que un gran número de componentes y que cada componente tiene una probabilidad no trivial de fracaso significa que algún componente de HDFS siempre es no funcional. Por lo tanto, la detección de fallas y la recuperación rápida y automática de los mismos es un objetivo central de arquitectura de HDFS. [5]

HDFS tiene una arquitectura maestro/esclavo. Un clúster HDFS consta de un solo NameNode, un servidor maestro que administra el espacio de nombres del sistema de archivos y regula el acceso a los archivos por clientes.

Además hay un número de DataNodes, generalmente uno por nodo en de clúster, que gestionan el almacenamiento de los nodos conectados que se ejecutan. Internamente, un archivo se divide en uno o más bloques y estos bloques se almacenan en un conjunto de DataNodes.

El NameNode ejecuta las operaciones de espacio de nombres de archivos del sistema, como archivos de apertura, cierre y cambio de nombre y directorios. También se determina la asignación de bloques para DataNodes.

Los DataNodes son responsables de atender las solicitudes de lectura y escritura de los clientes del sistema de archivos. Los DataNodes también llevan a cabo la creación de bloques, eliminación y duplicación en la instrucción de la NameNode. [5]

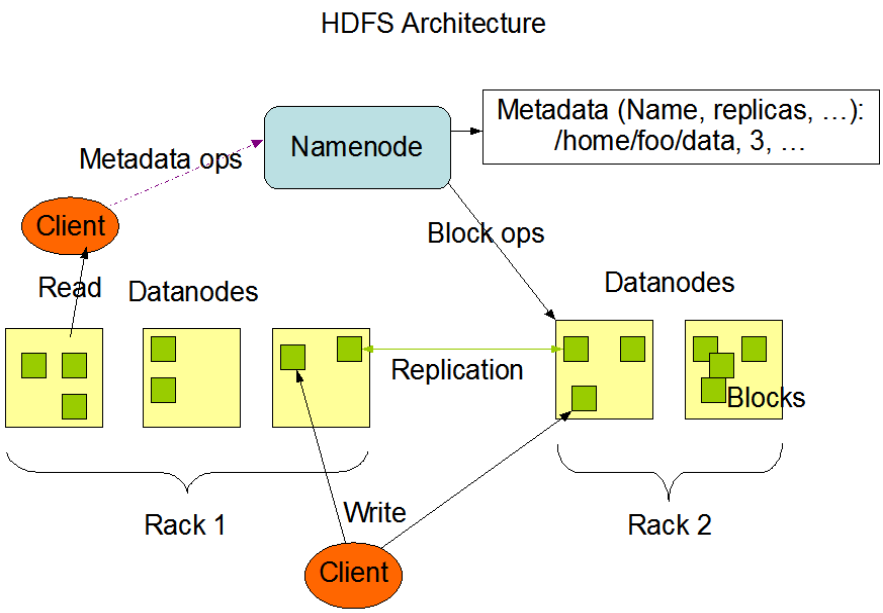


Figura 2-4 – Arquitectura HDFS [5]

NameNode y DataNode son piezas de software diseñado para ejecutarse en máquinas de los productos básicos. Estas máquinas suelen ejecutar un sistema operativo GNU/Linux. HDFS está construido utilizando el lenguaje Java, cualquier equipo que soporte Java puede ejecutar el NameNode o el software DataNode.

El NameNode es el árbitro y el repositorio de metadatos para todos los HDFS. [5]

2.1.3. Definición de MapReduce.

Es un framework de fácil creación de aplicaciones que procesan grandes cantidades de datos (multi-terabyte data-sets) en paralelo en grandes clúster (miles de nodos) de productos de hardware de una manera fiable y tolerante a fallos. [8]

MapReduce por lo general divide la entrada de un conjunto de datos en pedazos independientes, que son procesados por el mapa de tareas de una manera totalmente paralela.Normalmente los nodos de cálculo y los nodos de almacenamiento son los mismos, es decir, el framework y el sistema de archivos distribuidos MapReduce se ejecutan en el mismo conjunto de nodos. [8]

Esta configuración permite que el framework para programar tareas con eficacia en los nodos donde los datos ya está presente, lo que resulta un ancho de banda agregado muy alto en todo el clúster. [8]

El framework MapReduce consta de un solo maestro JobTracker y un esclavo TaskTracker por nodo del clúster. El maestro es responsable de programar las tareas de los puestos de trabajo

componente de los esclavos, su control y ejecución de los fallos de tareas. Los esclavos ejecutan las tareas según las condiciones del maestro. [8]

Como mínimo, se deben especificar las aplicaciones de entrada/salida y lugares de abastecimiento del mapa y reducir las funciones a través de implementaciones de interfaces apropiadas y/o clases abstractas y otros parámetros del Job, constituyen la configuración del Job. [8]

El Hadoop cliente del Job a continuación, envía el Job (jar/ejecutable, etc.) y la configuración del JobTracker que a su vez asume la responsabilidad de distribuir el software de configuración de los esclavos, la programación de las tareas y su respectivo seguimiento, proporcionando la información de estado y de diagnóstico para el job-client. [8]

El framework MapReduce opera exclusivamente con pares <clave,valor>, esto es, el framework ve la entrada del Job como un conjunto de pares <clave, valor>, y produce un conjunto de pares <clave,valor>, como la salida del Job, posiblemente de distintos tipos. [8]

2.2. Apache Pig.

Figura 2-5 – Logo de Pig [14]

Apache Pig provee un motor para la ejecución de flujo de datos en paralelo sobre Hadoop. Esto incluye un lenguaje de alto nivel, denominado Pig Latin, para expresar este flujo de datos. Pig Latin incluye operadores para muchas de las operaciones tradicionales de datos, como lo son join, sort, filter, etc. Así como la capacidad para que los usuarios desarrollen sus propias funciones para leer, procesar y escribir datos. [14]

Pig es un proyecto de código abierto de Apache. Esto significa que los usuarios son libres para descargarlo tanto como fuente o binarios, usarlos para ellos mismos, contribuir pero todo ello, bajo los términos de la Licencia de Apache.

Pig corre sobre la plataforma de Hadoop, el cual hace uso de las características de Hadoop Distributed File System, HDFS, y el sistema de procesamiento de Hadoop, denominado MapReduce.

HDFS es un sistema de archivos distribuido, que almacena archivos a través de los nodos en el clúster de hadoop. Este maneja la división de los archivos dentro de los bloques de Hadoop y distribuirlos a diferentes maquinas, incluyendo la replicación de data, así de esta manera si un máquina falla, la data no se pierda. Por defecto, Pig lee los archivos del HDFS, utiliza HDFS para almacenar data intermedia entre trabajos de MapReduce y escribe la salida hacia HDFS. [14]

CAPÍTULO 3

3. ANÁLISIS DE LA SOLUCIÓN.

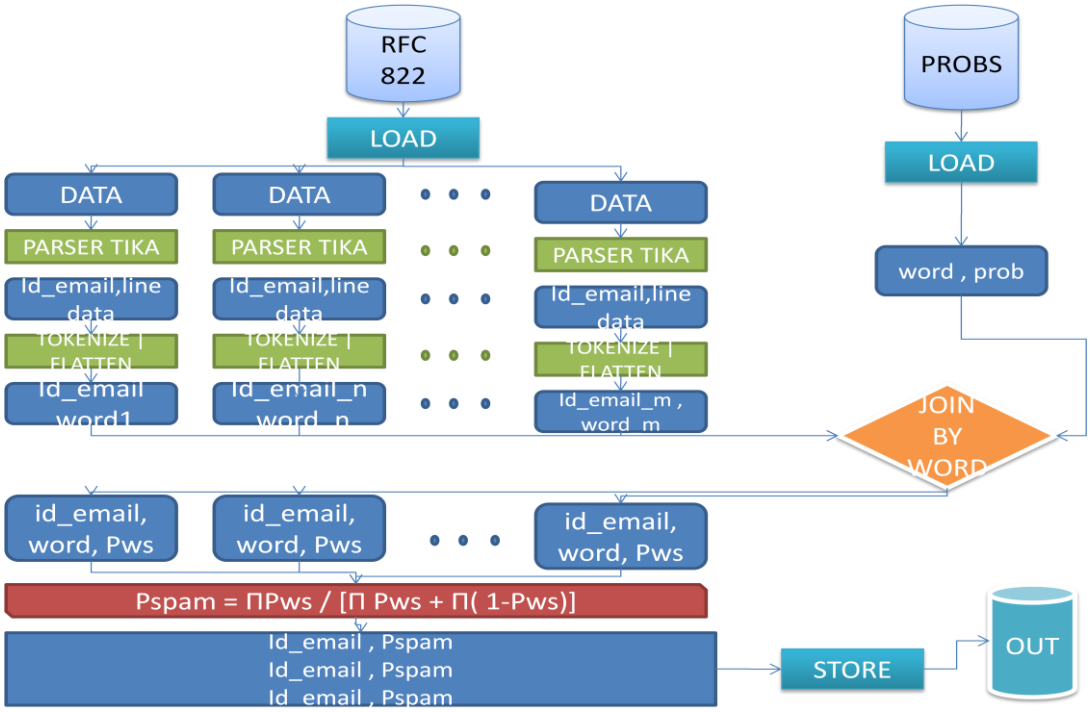


Figura 3-6 - Esquema de la solución [3]

La solución propuesta para la detección de mensajes spam, se basa en los siguientes componentes:

Parser TIKA.- Es el primer componente por el cual la data atraviesa dentro del programa Antispam, aquí ingresa el mensaje según el estándar RFC822 y el parser de Apache Tika extrae la meta data, encabezados y contenidos del mensaje para luego ser procesados por el filtro. [15]

Filtro (word, probabilidad-spam).- Este es el filtro bayesiano, previamente entrenado utilizando un corpus de mensajes de spam y otro corpus de mensajes no spam, con el cual se determina la probabilidad individual de que esta palabra en un mensaje sea un spam.

Formula de probabilidad.- Esta es la fórmula de probabilidad conjunta en la cual se suman las probabilidades individuales de las palabras contenidas en el mensaje, y se determina su porcentaje de probabilidad de que dicho mensaje sea un spam.

3.1. Requerimientos.

Tomando en consideración que la solución propuesta se centra en la construcción de un anti-spam con apache Hadoop mediante la utilización de las funcionalidades de MapReduce, se definen las características principales que este debe cumplir en las sub secciones a continuación.

3.1.1. Requerimientos Hardware.

- Computador (1 o más dependiendo del número de nodos y de la cantidad de información).

- Tarjeta de red (1 por cada computador).
- Memoria RAM de 4Gb o superior.
- Procesador de dos o más núcleos.
- Disco duro de capacidad considerable, recomendable 500Gb o superior.

3.1.2. Requerimientos Software.

- Sistema operativo de alguna de las distribuciones de Linux.
- Apache Hadoop
- Apache Pig
- Apache Tika
- Pig Latin
- Java 1.6 o superior
- SSH.

3.2. Análisis de las Capacidades del framework a utilizar.

Apache Hadoop nos permite mapear, examinar, analizar y comparar los datos almacenados en Hadoop por medio de sus funcionalidades asociadas y/o creadas por el desarrollador para obtener los resultados de mensajes spam. El incremento de data no afectara el rendimiento de la aplicación siempre y cuando así como crezcan los datos a

analizar, también se añadan nuevos nodos al clúster, tanto para que aumente la capacidad de almacenamiento así como la de la computación distribuida.

Apache Tika por el momento cuenta solamente con el parser para mensajes que cumplan el estándar RFC822, así que los datos ingresados por el momento solo se aceptaran si siguen este estándar, cabe recalcar que el core de Apache Tika, con cada nueva liberación, incluye nuevos parsers, y además como es de naturaleza código abierto permite a desarrolladores foráneos implementar y desarrollar nuevos parsers, si fueran necesario. Así que la limitación del formato de los datos de entrada, tan solo correspondería a adaptar un nuevo parser diseñado para Apache Tika. [15]

Apache Pig nos permite crear bajo su lenguaje de alto nivel, aplicaciones muy desarrolladas sin enfocarse en la programación de bajo nivel de mappers y reducers, aunque en algunos casos específicos se ha demostrado que la implementación directamente de programas sobre las clases map y reduce funciona con una mejor velocidad si se definen las clases Combiner y partitioner. [14]

Cabe recalcar que Apache Pig también proporciona su API para que desarrolladores aventureros integren sus clases personalizadas de Combiner y partitioner. Por lo cual también Apache Pig nos brinda un amplio rango de ventajas y hace que las limitaciones en el framework sean muchos menores.

3.3. Requerimientos y datos o mensajes correos generados o almacenados.

Los mensajes de entrada deben seguir el estándar RFC822, norma de ARPANET para mensajes electrónicos sobre Internet. Los cuales para nuestra aplicación han sido descargados de sitios con un corpus de mensajes libres para la investigación y desarrollo de aplicaciones relacionadas con el spam o el análisis de mensajes electrónicos. [16]

No se han generado mensajes artificiales, y cabe recalcar que toda la data corresponde a data real, proporcionada por empresas con corpus de mensajes disponibles para la sociedad en general.

CAPÍTULO 4

4. DISEÑO E IMPLEMENTACIÓN DE HADOOP PARA COMBATIR EL SPAM.

En este capítulo se describe la metodología usada para la implementación de Hadoop, Pig y de las funcionalidades de MapReduce para combatir el spam. Posteriormente se detalla el plan de pruebas utilizado.

4.1. Modelo Lógico del proceso de identificación de spam.

Creación de el Hashmap<word , probabilidad –spam>, con corpus de prueba con :

Mensajes spam: 2176
Mensajes no spam: 2490

bad<> = Hashmap word , cantidad veces(Corpus spam)
good<>= Hashmap word, cantidad veces(Corpus no spam)
B= cantidad mensajes spam
G=cantidad mensajes no spam

rg=relacion-good
rb=relacion-bad

Donde a partir de esta información se determina la probabilidad de spam individual de cada palabra.

```
rg = min(1, 2(good(w)/G)), rb = min(1, bad(w)/B)
Pspam|w = max(.01,min(.99, rb/(rg + rb)))
```

Con lo cual se consigue un Hashmap <word, probabilidad> y el cual es almacenado sobre HDFS en un archivo llamado probs dentro de /user/hadoop/

Análisis con corpus de producción:

Mensajes spam:	52799
Mensajes no spam:	32399

Los mensajes de entrada ya sean spam o no spam, son identificados con un id mensaje del cual se extrae la metada, encabezados, contenido y links en el cuerpo del mensaje.

Se realiza un join con el Hashmap<word, probabilidad-spam> para cada palabra y se identifica la probabilidad individual de cada palabra de todos los mensajes a la vez.

Cabe recalcar que las palabras que se encuentre en un mensaje y no se encuentre en el Hashmap de probabilidad tendrán directamente una probabilidad asignada de 0.4. Que es una buena elección después de los análisis de prueba y error, que en el ensayo “A plan for spam”, detalla más abiertamente. [17]

Con lo cual finalmente se calcula la probabilidad conjunta de un email a través de las probabilidades de las palabras que contiene. Utilizando la formula de probabilidad.

$$P_{spam} = \frac{\prod_{i=1}^{15} P_{spam|w_i}}{\prod_{i=1}^{15} P_{spam|w_i} + \prod_{i=1}^{15}(1 - P_{spam|w_i})}$$

4.2. Modelamiento de los Datos para su Procesamiento.

Los datos modelados para su procesamiento se compondrían básicamente de mensajes con el estándar RFC822, norma de ARPANET para mensajes de internet, los cuales tienen la metada siguiente:

```
Date     :  27 Aug 76 0932 PDT
From     :  Ken Davis <KDavis@This-Host.This-net>
Subject  :  Re: The Syntax in the RFC
Sender   :  KSecy@Other-Host
Reply-To :  Sam.Irving@Reg.Organization
To       :  George Jones <Group@Some-Reg.An-Org>,
                 Al.Neuman@MAD.Publisher
cc       :  Important folk:
                   Tom Softwood <Balsa@Tree.Root>,
                   "Sam Irving"@Other-Host;,
                 Standard Distribution:
                   /main/davis/people/standard@Other-Host,
                   "<Jones>standard.dist.3"@Tops-20-Host>;
Comment  :  Sam is away on business. He asked me to handle
             his mail for him.  He'll be able to provide  a
            more  accurate  explanation  when  he  returns
                 next week.
In-Reply-To: <some.string@DBM.Group>, George's message
X-Special-action:  This is a sample of user-defined field-
                 names.  There could also be a field-name
                                      "Special-action", but its name
                   might later  be    preempted
Message-ID: <4231.629.XYzi-What@Other-Host>
```

4.3. Plan de Pruebas.

Pruebas en modo stand alone en el Laboratorio de Ingeniería de Software.

Ejecutando Apache Hadoop, Apache Pig en una computadora del laboratorio de Software en el cual sobre una maquina corren todos los demonios, debido a que se había establecido una configuración semi-distribuida.

Host: wrks129-155fiec

NameNode

JobTracker

SecondaryNameNode

Tasktracker

DataNode

Pruebas con múltiples nodos en el Laboratorio de Ingeniería de Software.

Ejecutando Apache Hadoop, Apache Pig en el laboratorio de Software con varios computadores se realizaron dos pruebas detalladas a continuación:

Con tres maquinas:

Wrks129-155fiec NameNode, JobTracker, SecondaryName.

Wrks129-134fiec DataNode, TaskTracker.

Wrks129-135fiec DataNode, TaskTracker.

Con seis maquinas:

Wrks129-155fiec NameNode, JobTracker, SecondaryName.

Wrks129-134fiec DataNode, TaskTracker.

Wrks129-135fiec DataNode, TaskTracker.

Wrks129-136fiec DataNode, TaskTracker.

Wrks129-137fiec DataNode, TaskTracker.

Wrks129-142fiec DataNode, TaskTracker.

4.4. Implementación de Apache Hadoop, Apache Pig y MapReduce para combatir el spam.

4.4.1. Herramientas a Utilizarse.

Java para que funcione Hadoop, obligatoriamente se debe definir la variable de entorno JAVA_HOME

Apache Hadoop para almacenar los datos y levantar los demonios

Apache Pig utilizando su lenguaje de alto nivel Pig Latin para crear Jobs de Hadoop, sin preocuparnos en el desarrollo de bajo nivel de clases Mappers o Reducers.

4.4.2. Instalación y configuración de las herramientas a utilizarse.

Las especificaciones que se darán a continuación fueron realizadas bajo CentOS release 5.5.

Primeramente crear un usuario para Hadoop

```
$ useradd hadoop
$ passwd hadoop
```

Configuración de la ruta de ubicación del java.

Digitar en el terminal lo siguiente

```
$ gedit .bash_profile
```

Lo cual abre el archivo .bash_profile

Agregar lo siguiente

```
JAVA_HOME=/usr/lib/jvm/java-1.6.0-openjdk-1.6.0.0/jre
export JAVA_HOME
```

Guardar los cambios y cerrar.

Ver la ruta del JAVA_HOME digitando

```
$ echo $JAVA_HOME
```

Si no se visualiza la ruta digitar en el terminal

```
$ gedit .bashrc
```

Y realizar el mismo procedimiento.

Descargando hadoop-0.20.203.0

http://www.takeyellow.com/apachemirror//hadoop/core/hadoop-0.20.203.0/

Una vez obtenido el archivo hadoop-0.20.203.0rc1.tar.gz proceder a colocarlo en la raíz y descomprimirlo, luego acceder a la carpeta **conf** que se encuentra en la ruta **/hadoop-0.20.203.0/conf/** y editar el archivo **hadoop-env.sh**

```
$ cd /hadoop-0.20.203.0/conf/
$ gedit hadoop-env.sh
```

Descomentar la línea donde se encuentra el JAVA_HOME y poner la ruta del mismo quedando así.

```
export JAVA_HOME=/usr/lib/jvm/java-1.6.0-openjdk-
1.6.0.0/jre
```

Configurando en modo Stand alone [6]

Como se está configurando como usuario hadoop dar permisos de escritura a los archivos que se editarán.

```
$ su root
$ chmod 766 /hadoop-0.20.203.0/conf/corsite.xml
$ chmod 766 /hadoop-0.20.203.0/conf/hdfs-site.xml
$ chmod 766 /hadoop-0.20.203.0/conf/mapred-site.xml
```

Para mayor facilidad se recomienda tener dos terminales abiertas la primera como root para otorgar permisos y la segunda como hadoop para las configuraciones.

Editar el archivo corsite.xml

```
$ su hadoop
$ gedit /hadoop-0.20.203.0/conf/coresite.xml
```

Agregar lo siguiente, guardar y cerrar el archivo.

coresite.xml

```
<configuration>
     <property>
         <name>fs.default.name</name>
         <value>hdfs://localhost:9000</value>
     </property>
</configuration>
```

Editar el archivo hdfs-site.xml

```
$ gedit /hadoop-0.20.203.0/conf/hdfs-site.xml
```

Agregar lo siguiente, guardar y cerrar el archivo.

hdfs-site.xml

```
<configuration>
     <property>
         <name>dfs.replication</name>
         <value>1</value>
     </property>
</configuration>
```

Editar el archivo mapred-site.xml

```
$ gedit /hadoop-0.20.203.0/conf/mapred-site.xml
```

Agregar lo siguiente, guardar y cerrar el archivo.

mapred-site.xml

```
<configuration>
     <property>
         <name>mapred.job.tracker</name>
         <value>localhost:9001</value>
     </property>
</configuration>
```

Para evitar que se pida la contraseña para acceder vía ssh se debe configurar lo siguiente:

```
$ ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
$ chmod 700 ~/.ssh
$ chmod 600 ~/.ssh/id_dsa
$ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
$ chmod 700 ~/.ssh
$ chmod 600 ~/.ssh/authorized_keys
$ ssh localhost
```

Ejecución del nodo

```
$ /hadoop-0.20.203.0/bin/hadoop namenode -format
$ /hadoop-0.20.203.0/bin/start-all.sh
```

Para verificar que la configuración se realizó con éxito Acceda al NameNode y al JobTracker por una interfaz web

Name Node – http://localhost:50070/

Job Tracker – http://localhost:50030/

Configurando en modo Clúster (modo completamente distribuido) [7]

Después de enfatizar los beneficios del almacenamiento y computación distribuida. A continuación se detalla las configuraciones necesarias.

Para el uso de las maquinas server, se seguirá un patrón para el nombre de estos servers:

Máster.- El nodo maestro del clúster y que alojara a los demonios NameNode y JobTracker.

Backup.- El server que alojara el demonio Secondary NameNode. hadoop1, hadoop2, hadoop3... - Las máquinas esclavos del clúster que alojaran a los demonios DataNode y TaskTracker.

Usando la convención de nombre precedente, se detallara a continuación la configuración de los archivos XML, situados exactamente en la ruta $HADOOP_HOME/conf/.

Editar el archivo corsite.xml

```
$ su hadoop
$ gedit /hadoop-0.20.203.0/conf/corsite.xml
```

Agregar lo siguiente, guardar y cerrar el archivo.

coresite.xml

```
<configuration>
   <property>
     <name>fs.default.name</name>
     <value>hdfs://master:9000<value>
     <description>The name of the default file system.
A URI whose scheme and authority determine the File
System implementation. </description>
   </property>
</configuration>
```

Editar el archivo mapred-site.xml

```
$ gedit /hadoop-0.20.203.0/conf/mapred-site.xml
```

Agregar lo siguiente, guardar y cerrar el archivo.

mapred-**site.xml**

```
<configuration>
   <property>
      <name>mapred.job.tracker</name>
      <value>master:9001</value>
      <description> The host and port that the
MapReduce JobTracker runs at. </description>
   </property>
</configuration>
```

Editar el archivo hdfs-site.xml

```
$ gedit /hadoop-0.20.203.0/conf/hdfs-site.xml
```

Agregar lo siguiente, guardar y cerrar el archivo.

hdfs-site.xml

```
<configuration>
   <property>
     <name>dfs.replication</name>
     <value>3</value>
     <description> The actual number of replications
can be specified when the file is created.
</description>
   </property>
</configuration>
```

Finalmente también necesitamos actualizar los archivos máster y esclavos para reflejar las localizaciones de todos los demonios.

```
[hadoop-user@master]$ cat masters
backup
[hadoop-user@master]$ cat slaves
hadoop1
hadoop2
hadoop3
```

Una vez que los archivos anteriormente mencionados han sido copiados a través de todos los nodos en el clúster, se necesita formatear el HDFS para prepararlo para el almacenamiento.

```
[hadoop-user@master]$ bin/namenode -format
```

Ahora podemos iniciar los demonios de Hadoop.

```
[hadoop-user@master]$ bin/start-all.sh
```

Y luego verificar que sobre los nodos están corriendo sus demonios asignados.

```
[hadoop-user@master]$ jps
30879 JobTracker
30717 NameNode
30965 Jps

[hadoop-user@backup]$ jps
2099 Jps
1679 SecondaryName Node

[hadoop-user@hadoop1]$ jps
7101 TaskTracker
7617 Jps
6988 DataNode
```

Con esto ya tenemos la configuración del clúster Hadoop completa.

CAPÍTULO 5

5. PRUEBAS Y RESULTADOS.

5.1. Ejecución de las pruebas

Las pruebas se realizaron en dos ambientes: stand alone y multinodo. En cada ambiente se procesaron la misma cantidad de correos. Para la implementación se diseñó el script tesis.pig**.** Esta script tiene como objetivo determinar la probabilidad de spam de cada correo a evaluar.

En stand alone se realizó una prueba con los 52790 spam y 32990.

En multinodo se realizaron dos pruebas, la primera con 6 computadores, 5 como DataNodes y una como NameNode para lo cual configuramos 1 como NameNode, JobTracker y los 4 restantes como DataNode, TaskTracker, la segunda con 4 computadores, 3 como DataNodes y una como NameNode para lo cual configuramos 1 como NameNode, JobTracker y los 3 restantes como DataNode, TaskTracker.

(Ver los detalles de las pruebas en los anexos)

Prueba 1

Tiempo de procesamiento			0:51:23
Correos procesados		**# Cantidad**	**Detectados como Spam**
Correos Spam		52799	
Correos No Spam		32990	
Total		92189	48136

Tabla I – Resultados prueba # 1

Filtro aplicado a correos spam(Falsos Negativos)			
Cantidad	**# Detectados como Spam**	**Porcentaje de Eficiencia**	**Porcentaje de Falsos Negativos**
52790	46981	88.99%	11.01%
Total		88.99%	11.01 %

Tabla II – Eficiencia Filtro| Falsos Negativos # 1

Filtro aplicado a correos no spam(Falsos Positivos)			
Cantidad	**# Detectados como Spam**	**Porcentaje de Eficiencia**	**Porcentaje de Falsos Positivos**
32990	1155	96.49%	3.51%
Total		96.49%	3.51 %

Tabla IIII – Eficiencia Filtro| Falsos Positivos # 1

Prueba 2

Tiempo de procesamiento			0:29:56
Correos procesados		**# Cantidad**	**Detectados como Spam**
Correos Spam		52799	
Correos No Spam		32990	
Total		92189	48136

Tabla IVV – Resultados prueba # 2

Filtro aplicado a correos spam(Falsos Negativos)			
Cantidad	**# Detectados como Spam**	**Porcentaje de Eficiencia**	**Porcentaje de Falsos Negativos**
52790	46981	88.99%	11.01%
Total		88.99%	11.01 %

Tabla V – Eficiencia Filtro| Falsos Negativos # 2

Filtro aplicado a correos no spam(Falsos Positivos)			
Cantidad	**# Detectados como Spam**	**Porcentaje de Eficiencia**	**Porcentaje de Falsos Positivos**
32990	1155	96.49%	3.51%
Total		96.49%	3.51 %

Tabla VI – Eficiencia Filtro| Falsos Positivos # 2

Prueba 3

Tiempo de procesamiento			0:22:35
Correos procesados		**# Cantidad**	**Detectados como Spam**
Correos Spam		52799	
Correos No Spam		32990	
Total		92189	48136

Tabla VII – Resultados prueba # 3

Filtro aplicado a correos spam(Falsos Negativos)			
Cantidad spam	**# Detectados como Spam**	**Porcentaje de Eficiencia**	**Porcentaje de Falsos Negativos**
52790	46981	88.99%	11.01%
Total		88.99%	11.01 %

Tabla VIII – Eficiencia Filtro| Falsos Negativos # 3

Filtro aplicado a correos no spam(Falsos Positivos)			
Cantidad no spam	**# Detectados como Spam**	**Porcentaje de Eficiencia**	**Porcentaje de Falsos Positivos**
32990	1155	96.49%	3.51%
Total		96.49%	3.51 %

Tabla IX – Eficiencia Filtro| Falsos Positivos # 3

5.2. Análisis de los resultados

General

Por los datos obtenidos en las pruebas, los cuales podemos revisar en las tablas de las tres pruebas donde lo que cambia son la cantidad de maquinas esclavo, se puede verificar que el porcentaje de eficacia del filtro para detectar correos spam es del 88.99%, esto de acuerdo a el análisis de aplicar el filtro únicamente a los mensajes spam, para correlacionar las pruebas totales contra los datos reales contenidos en los corpus, mientras que el porcentaje de falsos negativos asciende a 11.01% lo que representa que toda esta cantidad de mensajes electrónicos no fueron filtrados, por alguna razón y pasaron la verificación del filtro, así como vemos disminuido el porcentaje de falsos positivos a solo 3.51%, el cual representa un valor bastante aceptable, para ser un filtro entrenado por aproximadamente unos 5000 mensajes electrónicos.

En el siguiente gráfico se muestran los tiempos de duración de cada una de las pruebas realizadas con lo cual podemos visualizar que la prueba 3 tuvo un menor tiempo de duración debido a que se utilizo una mayor cantidad de nodos y el procesamiento paralelo tuvo éxito.

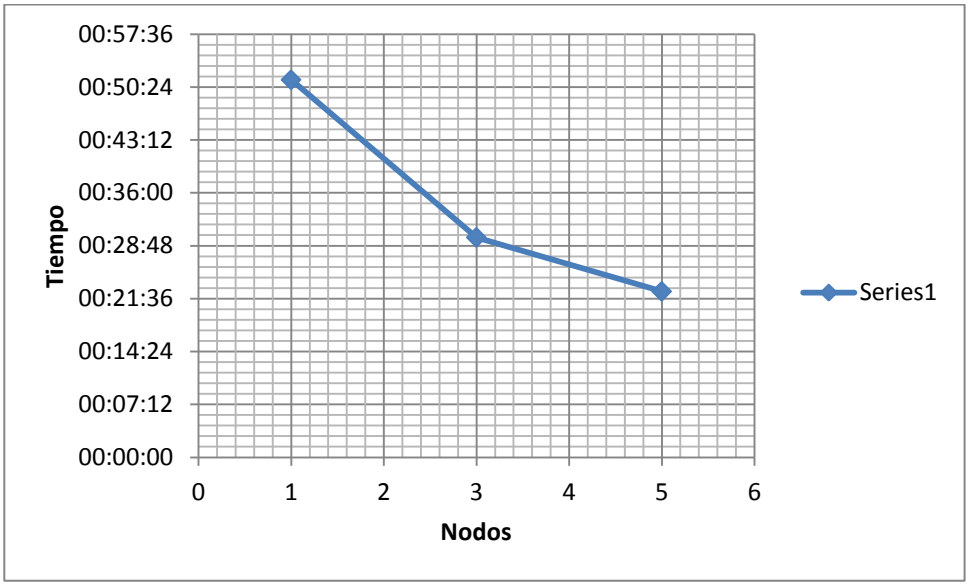


Figura 5-7 – Comparación de la Duración contra la cantidad de nodos

Entonces por regresión lineal para calcular aproximadamente la curva del tiempo en función de la cantidad de nodos, tendríamos lo siguiente:

$$\boldsymbol{\gamma = \beta 0 + \beta 1 x}$$

$$\boldsymbol{\beta 1 = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum(x^2) - (\sum x)^2}}$$

$$\boldsymbol{\beta 0 = \frac{\sum y - \beta 1(\sum x)}{n}}$$

Si n = 3 | Numero de pruebas

x	y	xy	x^2
0	∞	∞	∞
1	0:51:23 | 51.38	51.38	1
3	0:29:56 | 29.94	89.82	9
5	0:22:35 | 22.58	112.90	25
$\sum x = 9$	$\sum y = 103.9$	$\sum xy = 254.20$	$\sum x^2 = 35$

$$\beta 1 = \frac{3(254.20) - (9)(103.9)}{3(35) - (81)} = -7.1875$$

$$\boldsymbol{\beta 0 = \frac{103.9 - (-7.1875)(9)}{3} = 56.1958}$$

Entonces la variable dependiente tiempo en función de la cantidad de nodos según la regresión lineal seria:

$$\gamma = 56.1958 - 7.1875x$$

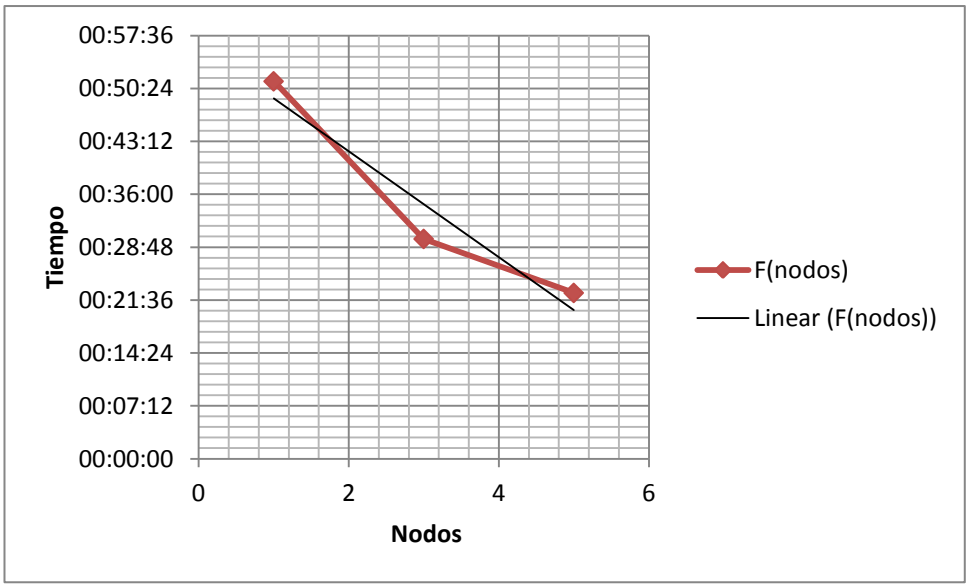


Figura 5-8 – Representación de la ecuación de regresión lineal.

Los siguientes 3 gráficos podemos visualizar el porcentaje de spam encontrado en cada una de las pruebas.

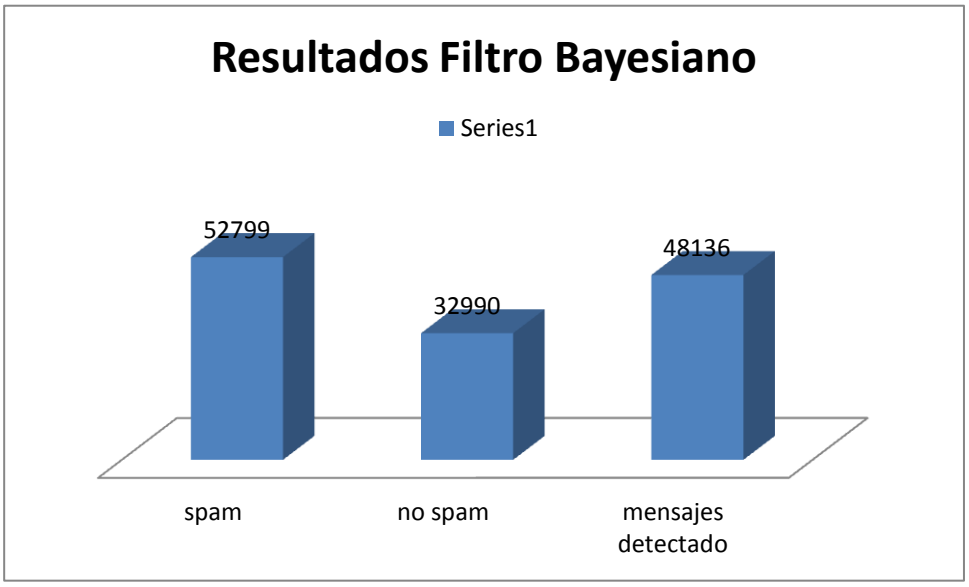


Figura 5-9 – Cantidad de correos detectados | Corpus real – Prueba

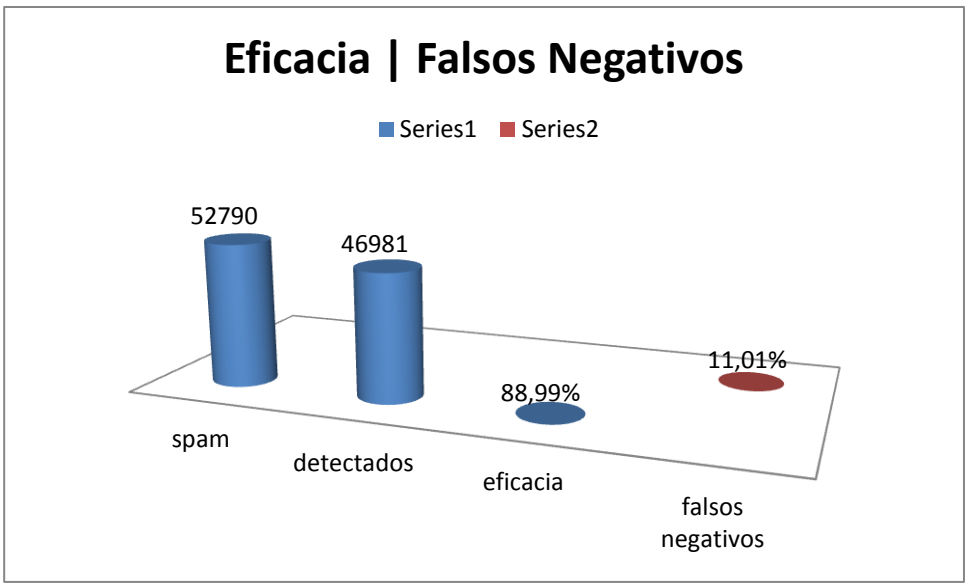


Figura 5-10 – Porcentaje de eficacia filtro | Falsos negativos – Prueba

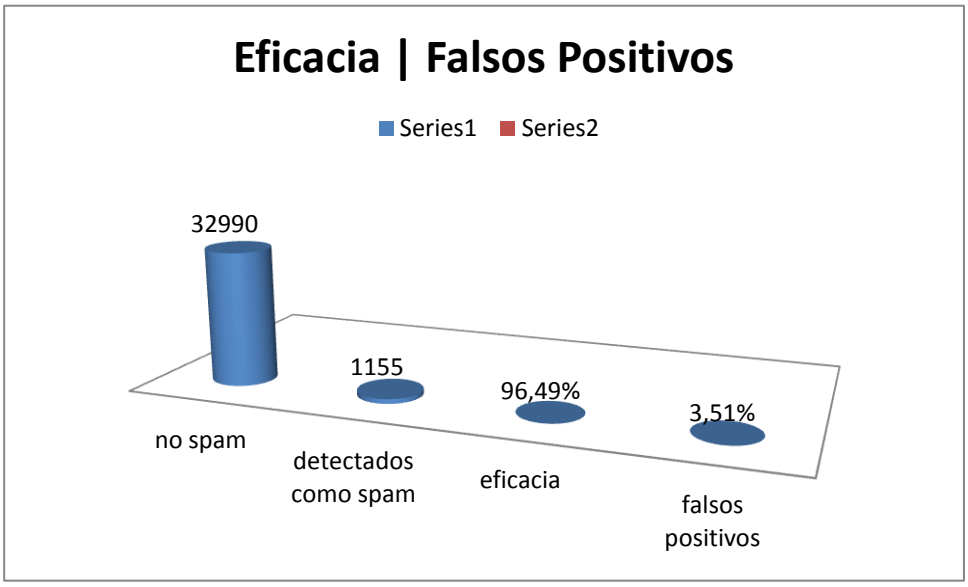


Figura 5-11 – Porcentaje de eficacia filtro |Falsos positivos – Prueba

CONCLUSIONES

Las conclusiones son:

1. Se ha determinado que el uso de filtros bayesianos utilizando un corpus de entrada mínimo para entrenar al filtro con aproximadamente 4000 mensajes entre mensajes spam y no spam tienes resultados bastante aceptables que ascienden a un 88.99% en un corpus de aproximadamente 85780.

2. El grado de probabilidad para que un mensaje sea considerado mensaje spam es que la probabilidad combinada basada en la fórmula del filtro es que sea mayor o igual al 90% lo que nos permite modificar estos resultados si así desea, dejándole un sesgo mayor o menor si así se desea.

3. Lo mejor a través del tiempo en esta clase de filtros bayesianos aplicados a la detección de spam es que no sería suficiente para los spammers hacer sus mensajes spam únicos o sencillamente dejar de utilizar palabras sucias que fácilmente sean detectadas, sino que aun tener que mantenerse cambiando el contenido del mensaje en cada uno de ellos y aun así si lo que desean es apuntar hacia un link, este si no es detectado a la primera vez, debido a la retroalimentación,

seria detectado en sucesivas oportunidades para lo cual quedaría registrado este sitio, y ya sea que cada día tendrían que registrar nuevos sitios que no hayan sido registrados porque sencillamente de otra manera el filtro spam lo detectaría sin problemas.

4. La herramienta de Base que se ha utilizado fue Hadoop de Apache debido a que los requisitos, parámetros de entrada y condiciones así lo requirieron, basándonos en su arquitectura maestro – esclavo , y su sistema de archivos distribuidos, HDFS, los cuales representan una poderosa herramienta tanto para tratar gran cantidad de datos o data masiva como es nuestro caso, como la tolerancia a fallos tanto para Hardware como para Software, debido a que para el caso de uso real, es necesario tener la posibilidad de agregar más espacio para la nueva data, basta con tan solo agregar un nuevo nodo o terminal, pa contar con el almacenamiento y capacidades de este nuevo nodo.

5. Además de contar con la replicación de la data lo cual no afectaría el workflow en tiempo real, si un terminal fallase por cualquier razón, el clúster y el JobTracker junto con el NameNode se basaran en la ubicación de los archivos de datos o bloques de datos replicados, para no afectar el rendimiento, ni resultados finales de la data, condición completamente importante para el análisis de resultados.

6. Debido a esto y las condiciones anteriormente expuestas, fue necesario para nosotros una herramienta que nos proporcione tolerancia a fallos, capacidad de almacenamiento masivo de datos, pero que a su vez, esto no disminuya su tiempo de respuesta, replicación de datos y un gran soporte detrás de esto, como lo proporciona Apache. Por esto y más, nuestra opción fue Apache Hadoop.

RECOMENDACIONES

Las recomendaciones son:

1. El filtro necesita un entrenamiento previo para la primera detección de spam de preferencia con una cantidad considerable de correos, luego de este entrenamiento será capaz de determinar por sí solo si un correo es legítimo o spam.

2. Mantener el grado de probabilidad en valores cercanos al 90% sin embargo se puede aumentar o disminuir el sesgo según se requiera.

3. Trabajar con varios nodos para prevención de fallos, mayor capacidad de almacenamiento y un tiempo de respuesta más optimo.
4. El clúster de computadoras o data center sobre el cual va a correr la plataforma Hadoop, debe ser analizada y revisada previamente por un especialista en redes, donde no se encuentre perdida de datos, ni ninguna clase de error en el envió de paquetes de una maquina a otra, debido que cada máquina ya sea configurado como maestro o esclavo, va a comunicarse directa o indirectamente con cada una de

los otros host, y esto ocasiona un error fatal, a la hora de el traspaso de data entre hosts.

ANEXOS

ANEXO A

From darksitmayder@miner.com Mon Jul 5 23:46:19 2007
Received: (from darksitmayder@localhost) by miner.com (8.9.3/8.9.3) id A05394;
Mon, 5 Jul 2007 23:46:18 -0500
Received: from staff.uiuc.edu (staff.uiuc.edu [128.174.5.59])
by miner.com (8.9.3/8.9.3) id XAA24214;
Mon, 5 Jul 2007 23:46:25 -0500
Date: Mon, 5 Jul 2007 23:46:18 -0500
From: John Doe <darksitmayder@miner.com>
To: John Smith <jsmith@staff.uiuc.edu>
MessagId: <200707052346.LAA05394@miner.com>
Subject: Hello there..!

hello jsmith@staff.uiuc.edu what have you been up to? Do you want to make some money?
http://t.co/8hG3o3BH Just check out this news article! I thank God that I found this opportunity and for my success!

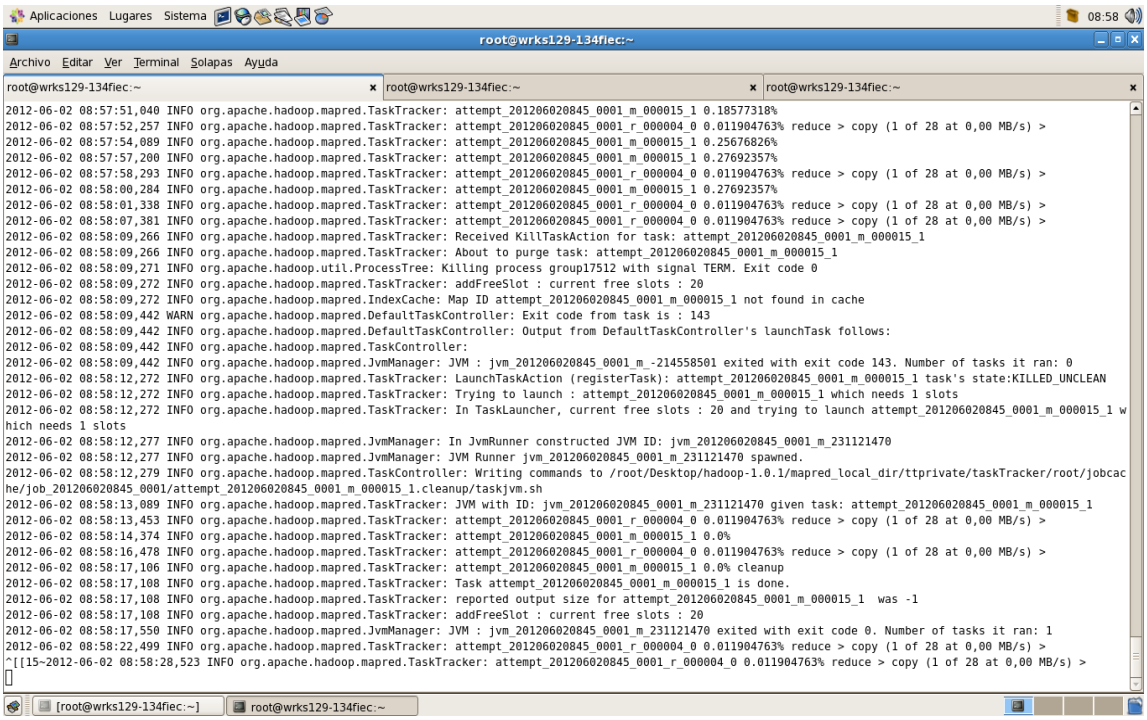
--_2e232f11-0d8d-45eb9fc-8bfc94631c7b_

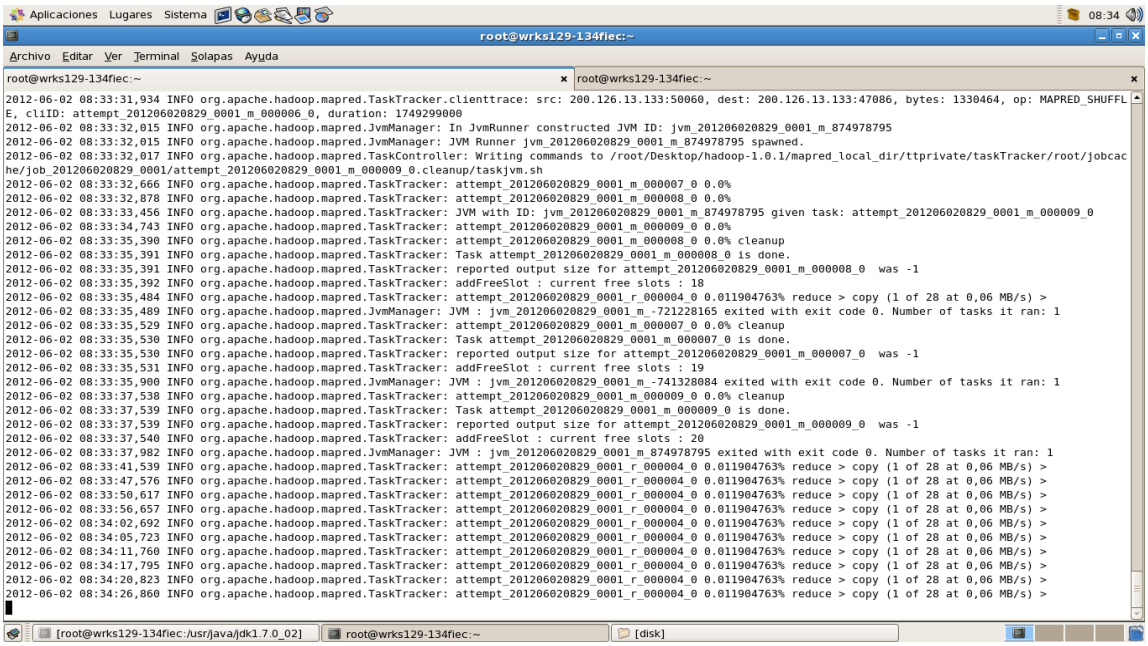
Content-Type: application/mspalabra
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="link.doc"

0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAAAAAAAAACAAAA1QAAAAAAAAAAA
EAAA2AAAAAEAAAD+////AAAAANMAAADUAAAA//
//
//
//
//
//
//
///s
pcEAA4AJBAAA+BK/AAAAAAAAEAAAAAAACAAAniYAAA4AYmpiarRWtFYAAAAAAAAAAAAAAAAAAAAA
AAAKDBYANDwAANY8AQDWPAEAch4AAAAAAAArAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAD//w8AAAAA
AAAAAAD//w8AAAAAAAAAAAAD//w8AAAAAAAAAAAAAAAAAAAAAAAALcAAAAAAF4nAAAAAAAAXicAAKU0
AAAAAAAApTQAAAAAAAAClNAAAAAAAAAKU0AAAAAAAApTQAABQAAAAAAAAAAAAAAAP////8AAAAAuTQA
AAAAAAC5NAAAAAAAAALk0AAA4AAAA8TQAADQAAAAlNQAAJAAAALk0AAAAAAAAA+bkAAPgCAABJNQAA
KAAAAHE1AAAWAAAAhzUAAAAAAAACHNQAAAAAAAIc1AAAAAAAArToAAAYBAACzOwAATAAAAP87AAAo
AAAAeLkAAAIAAAB6uQAAAAAAAHq5AAAAAAAAerkAAAAAAAB6uQAAAAAAAHq5AAAAAAAAerkAACQA
AADxvAAAogIAAJO/AAA+AAAAnrkAABUAAAAAAAAAAAAAAAAAAAAAAAAAApTQAAAAAAAAnPAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAABPOgAAXgAAAK06AAAAAAAAJzwAAAAAAAAnPAAAAAAAAJ65AAAAAAAA
AAAAAAAAAAClNAAAAAAAAAKU0AAAAAAAAhzUAAAAAAAAAAAAAAAAAAIc1AADIBAAAs7kAABYAAABF
QgAAAAAAAAEVCAAAAAAAARUIAAAAAAAAnPAAA9gQAAKU0AAAAAAAAhzUAAAAAAAAClNAAAAAAAAAIc1
AAAAAAAAeLkAAAAAAAAAAAAAAAAAEVCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAJzwAAAAAAAB4uQAAAAAAAAAAAAAAAAAAAARUIAAAAAAAABFQgAA
fgQAAOiiAADUAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATqoAAAAAAAACHNQAAAAAAAAP////8AAAAAAAMFu7Hd6
zAEAAAAAAAAAAALk0AAAAAAAAAHUEAAGoAAAC8pQAA2gAAAAAAAAAAAAAAAZLkAABQAAADJuQAAMAAA
APm5AAAAAAAAAlqYAALgDAADRvwAAAAAAAAIdBAAC+AAAA0b8AALAAAABOqgAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABO
qgAAYgMAANG/AAAAAAAAAAAAAAAAAAAAClNAAAAAAAAAALCtAAC0CwAAAAAAAAAAAAAAAAAAAAAAAAAEVC
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAJzwA
AAAAAAAnPAAAAAAAAACc8AAAAAAAAAnrkAAAAAAAACeuQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAARUIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACc8AAAA
--_2e232f11-0d8d-45eb9fc-8bfc94631c7b_--

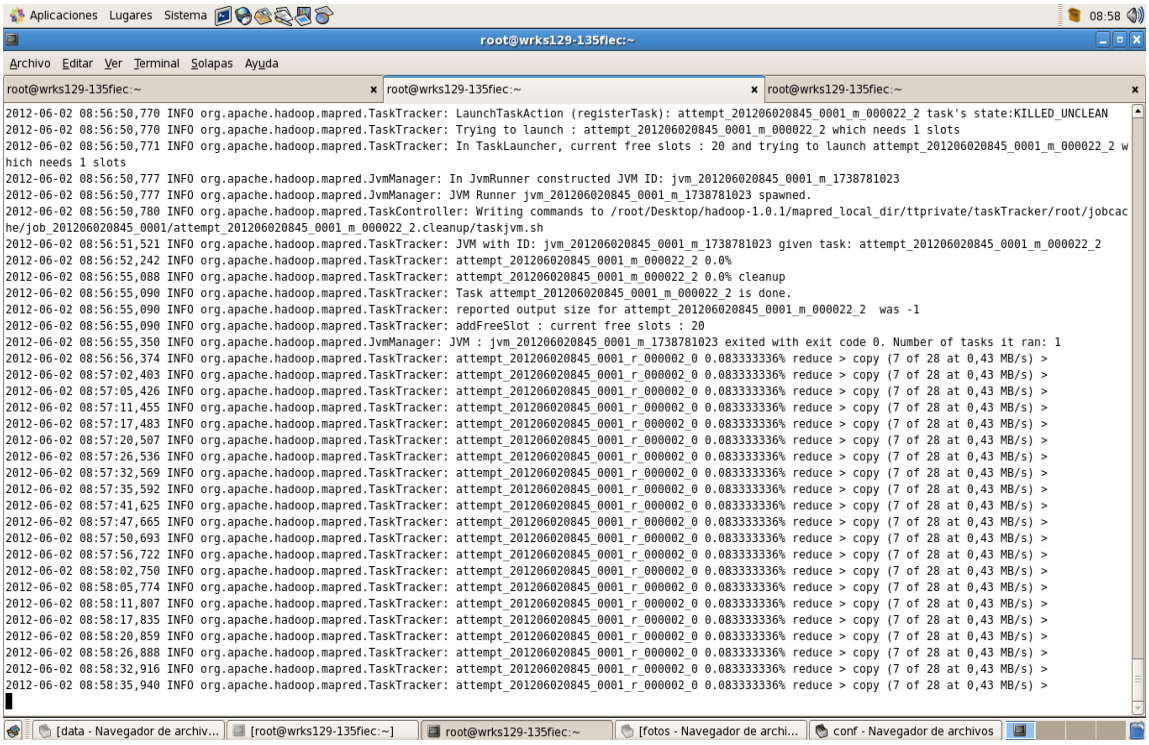
ANEXO B

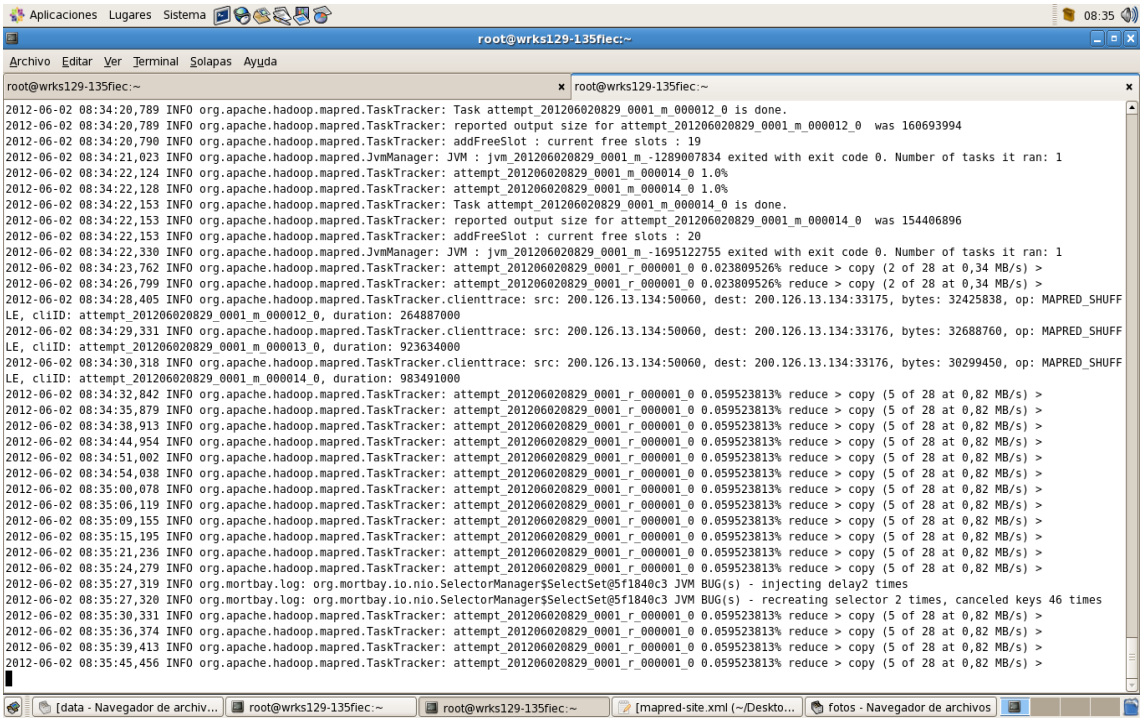
wrks129-134fiec

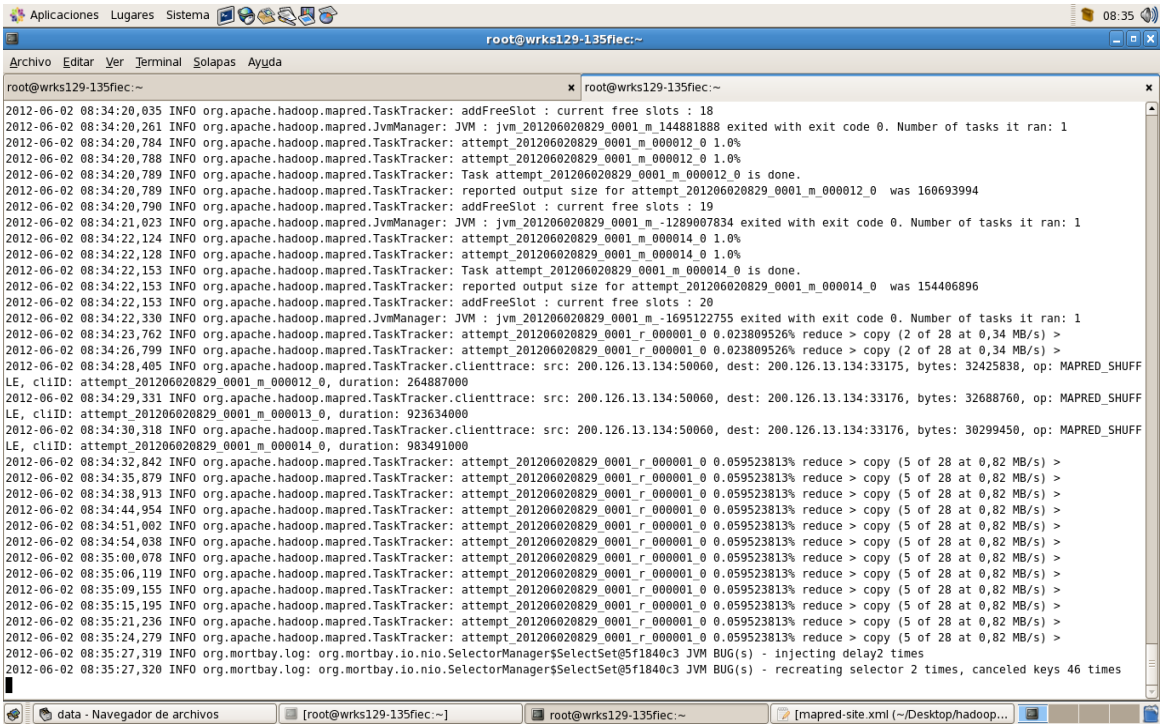




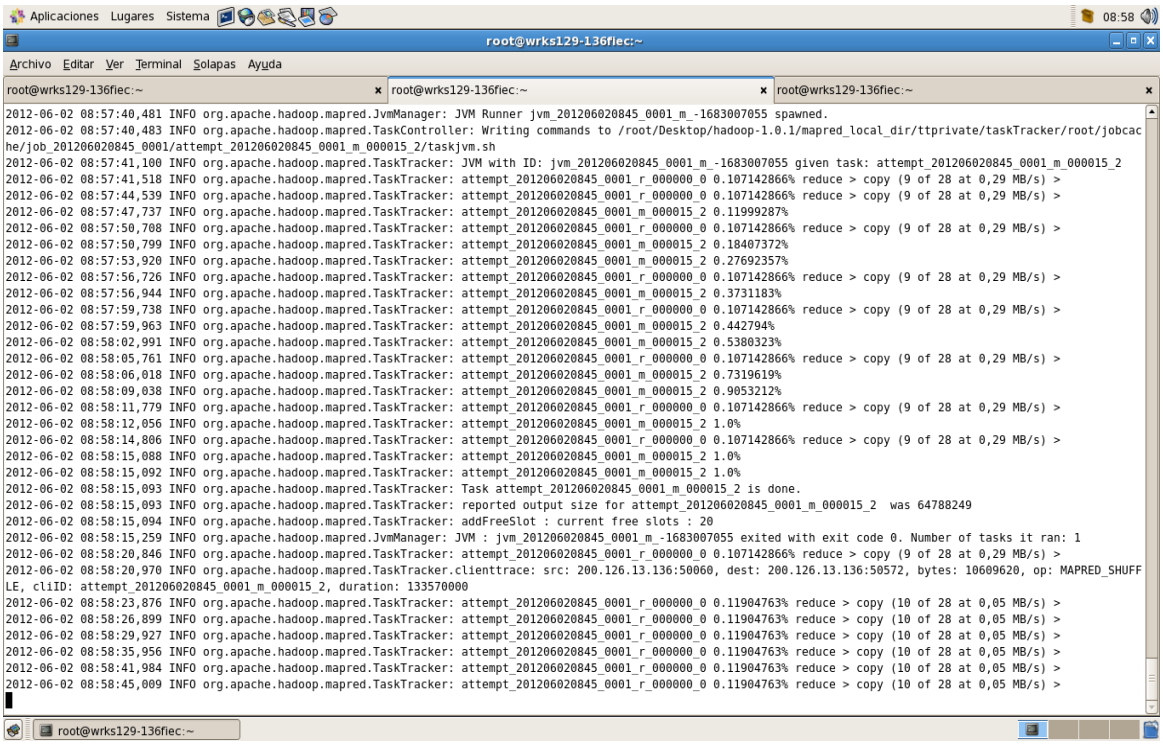
wrks129-135fiec

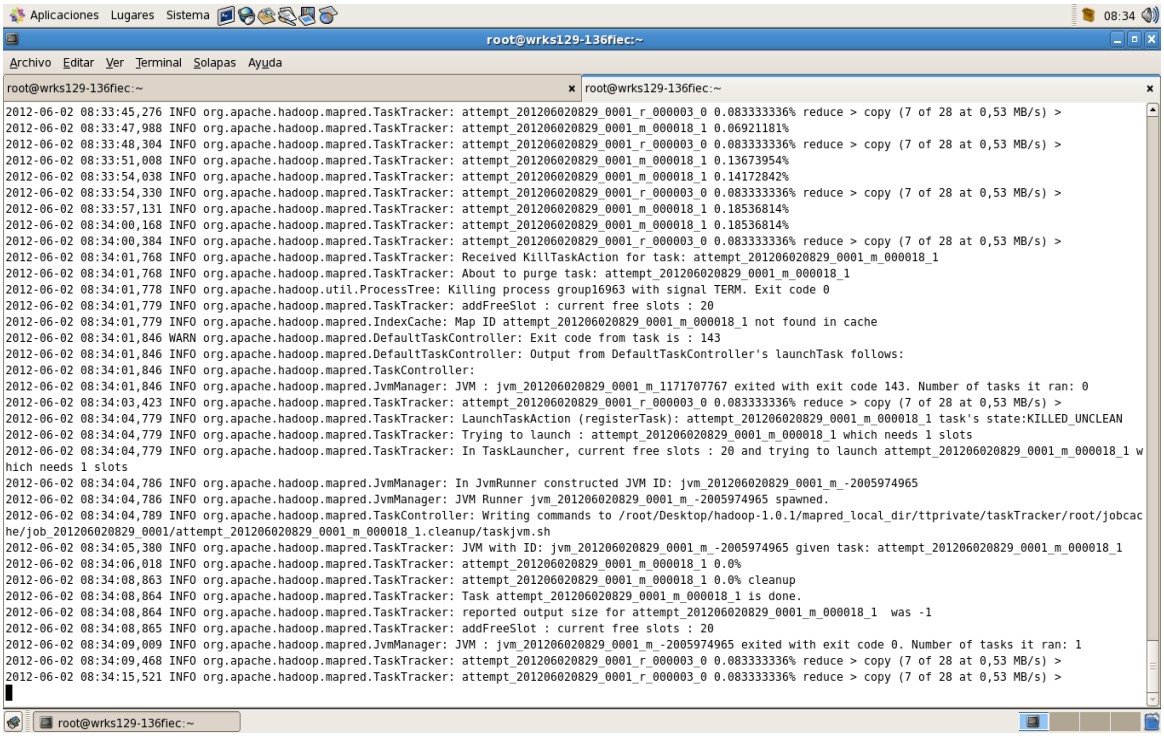


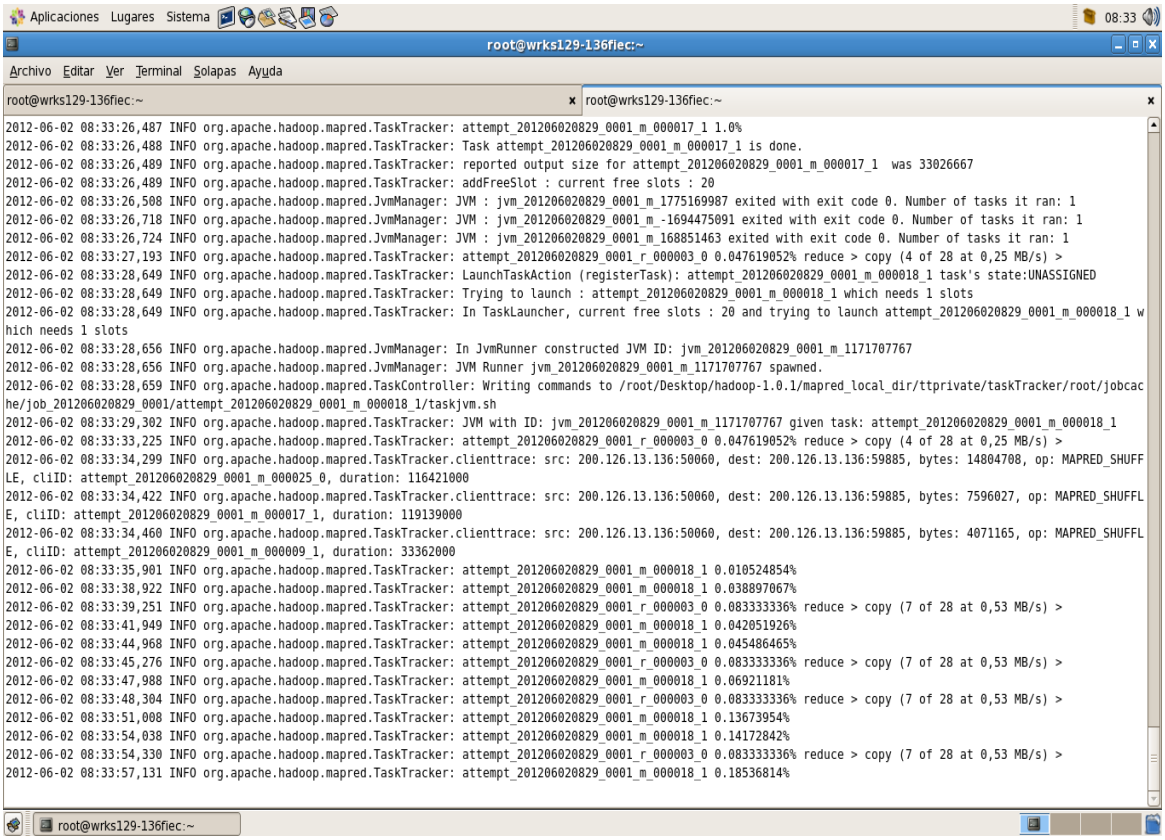




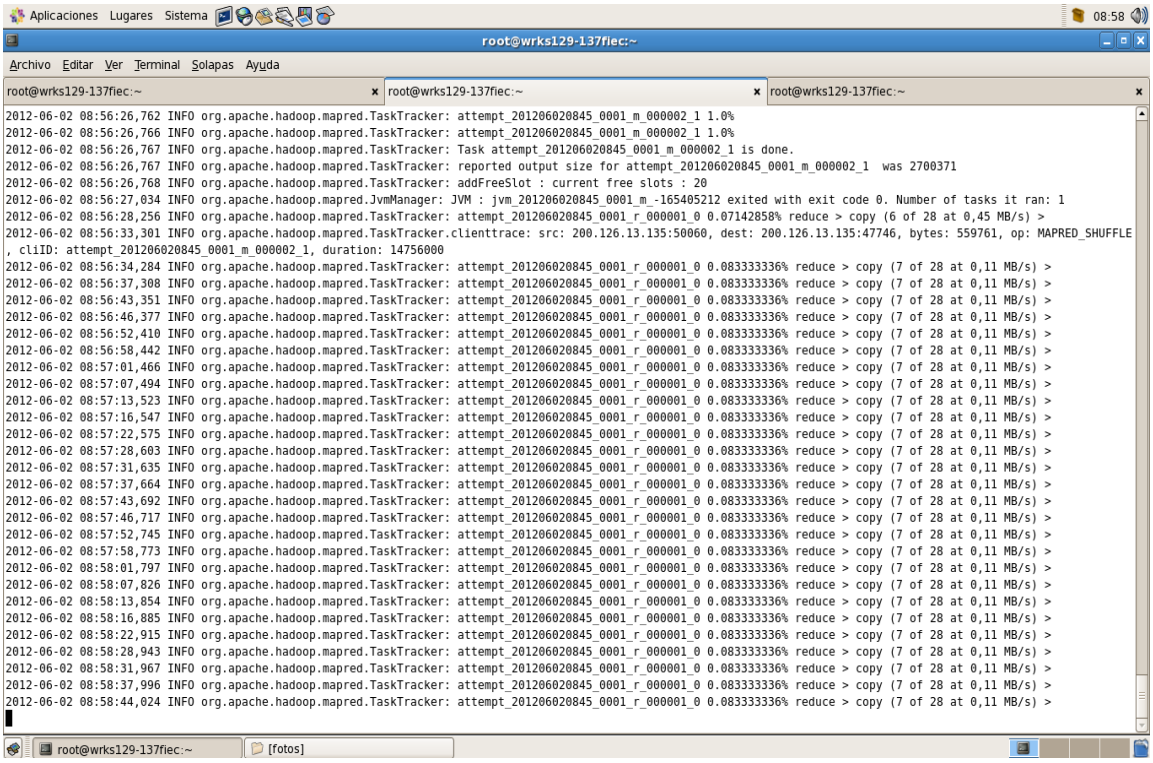
wrks129-136fiec

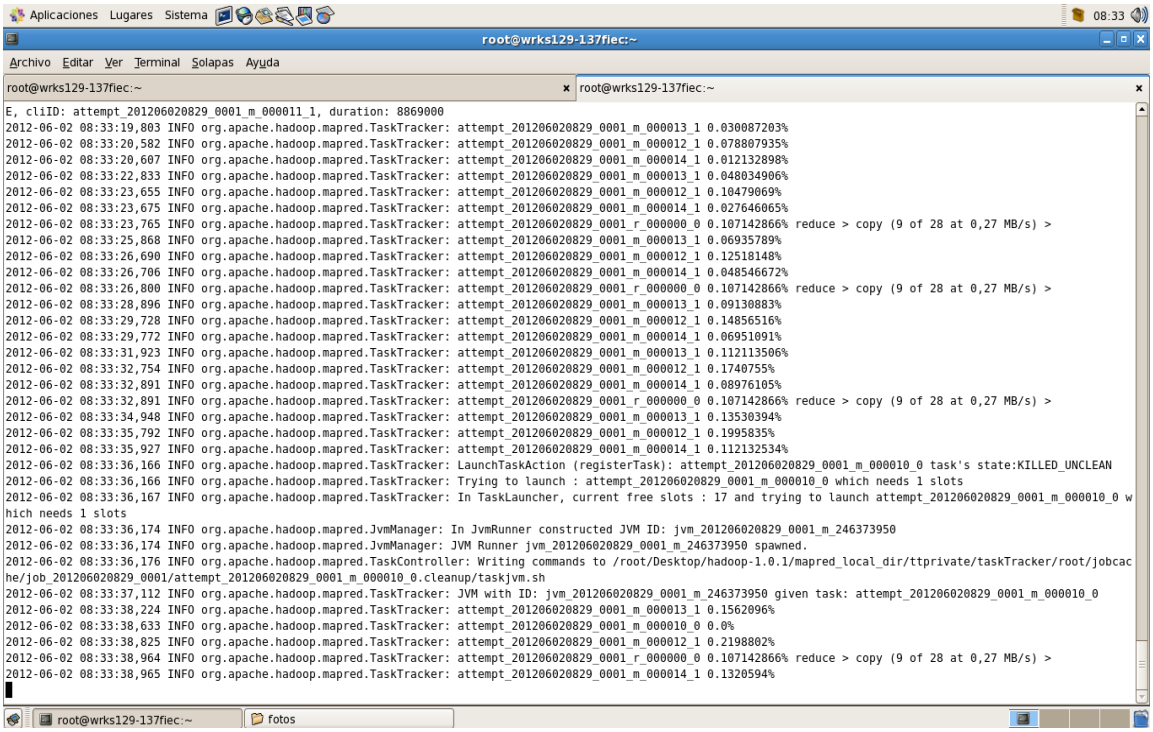



Aplicaciones Lugares Sistema
08:34
root@wrks129-136fiec:~
Archivo Editar Ver Terminal Solapas Ayuda
root@wrks129-136fiec:~
root@wrks129-136fiec:~
2012-06-02 08:33:45,276 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_r_000003_0 0.083333336% reduce > copy (7 of 28 at 0,53 MB/s) >
2012-06-02 08:33:47,988 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_m_000018_1 0.06921181%
2012-06-02 08:33:48,304 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_r_000003_0 0.083333336% reduce > copy (7 of 28 at 0,53 MB/s) >
2012-06-02 08:33:51,008 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_m_000018_1 0.13673954%
2012-06-02 08:33:54,038 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_m_000018_1 0.14172842%
2012-06-02 08:33:54,330 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_r_000003_0 0.083333336% reduce > copy (7 of 28 at 0,53 MB/s) >
2012-06-02 08:33:57,131 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_m_000018_1 0.18536814%
2012-06-02 08:34:00,168 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_m_000018_1 0.18536814%
2012-06-02 08:34:00,384 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_r_000003_0 0.083333336% reduce > copy (7 of 28 at 0,53 MB/s) >
2012-06-02 08:34:01,768 INFO org.apache.hadoop.mapred.TaskTracker: Received KillTaskAction for task: attempt_201206020829_0001_m_000018_1
2012-06-02 08:34:01,768 INFO org.apache.hadoop.mapred.TaskTracker: About to purge task: attempt_201206020829_0001_m_000018_1
2012-06-02 08:34:01,778 INFO org.apache.hadoop.util.ProcessTree: Killing process group16963 with signal TERM. Exit code 0
2012-06-02 08:34:01,779 INFO org.apache.hadoop.mapred.TaskTracker: addFreeSlot : current free slots : 20
2012-06-02 08:34:01,779 INFO org.apache.hadoop.mapred.IndexCache: Map ID attempt_201206020829_0001_m_000018_1 not found in cache
2012-06-02 08:34:01,846 WARN org.apache.hadoop.mapred.DefaultTaskController: Exit code from task is : 143
2012-06-02 08:34:01,846 INFO org.apache.hadoop.mapred.DefaultTaskController: Output from DefaultTaskController's launchTask follows:
2012-06-02 08:34:01,846 INFO org.apache.hadoop.mapred.TaskController:
2012-06-02 08:34:01,846 INFO org.apache.hadoop.mapred.JvmManager: JVM : jvm_201206020829_0001_m_1171707767 exited with exit code 143. Number of tasks it ran: 0
2012-06-02 08:34:03,423 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_r_000003_0 0.083333336% reduce > copy (7 of 28 at 0,53 MB/s) >
2012-06-02 08:34:04,779 INFO org.apache.hadoop.mapred.TaskTracker: LaunchTaskAction (registerTask): attempt_201206020829_0001_m_000018_1 task's state:KILLED_UNCLEAN
2012-06-02 08:34:04,779 INFO org.apache.hadoop.mapred.TaskTracker: Trying to launch : attempt_201206020829_0001_m_000018_1 which needs 1 slots
2012-06-02 08:34:04,779 INFO org.apache.hadoop.mapred.TaskTracker: In TaskLauncher, current free slots : 20 and trying to launch attempt_201206020829_0001_m_000018_1 w
hich needs 1 slots
2012-06-02 08:34:04,786 INFO org.apache.hadoop.mapred.JvmManager: In JvmRunner constructed JVM ID: jvm_201206020829_0001_m_-2005974965
2012-06-02 08:34:04,786 INFO org.apache.hadoop.mapred.JvmManager: JVM Runner jvm_201206020829_0001_m_-2005974965 spawned.
2012-06-02 08:34:04,789 INFO org.apache.hadoop.mapred.TaskController: Writing commands to /root/Desktop/hadoop-1.0.1/mapred_local_dir/ttprivate/taskTracker/root/jobcac
he/job_201206020829_0001/attempt_201206020829_0001_m_000018_1.cleanup/taskjvm.sh
2012-06-02 08:34:05,380 INFO org.apache.hadoop.mapred.TaskTracker: JVM with ID: jvm_201206020829_0001_m_-2005974965 given task: attempt_201206020829_0001_m_000018_1
2012-06-02 08:34:06,018 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_m_000018_1 0.0%
2012-06-02 08:34:08,863 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_m_000018_1 0.0% cleanup
2012-06-02 08:34:08,864 INFO org.apache.hadoop.mapred.TaskTracker: Task attempt_201206020829_0001_m_000018_1 is done.
2012-06-02 08:34:08,864 INFO org.apache.hadoop.mapred.TaskTracker: reported output size for attempt_201206020829_0001_m_000018_1 was -1
2012-06-02 08:34:08,865 INFO org.apache.hadoop.mapred.TaskTracker: addFreeSlot : current free slots : 20
2012-06-02 08:34:09,009 INFO org.apache.hadoop.mapred.JvmManager: JVM : jvm_201206020829_0001_m_-2005974965 exited with exit code 0. Number of tasks it ran: 1
2012-06-02 08:34:09,468 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_r_000003_0 0.083333336% reduce > copy (7 of 28 at 0,53 MB/s) >
2012-06-02 08:34:15,521 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_r_000003_0 0.083333336% reduce > copy (7 of 28 at 0,53 MB/s) >
root@wrks129-136fiec:~

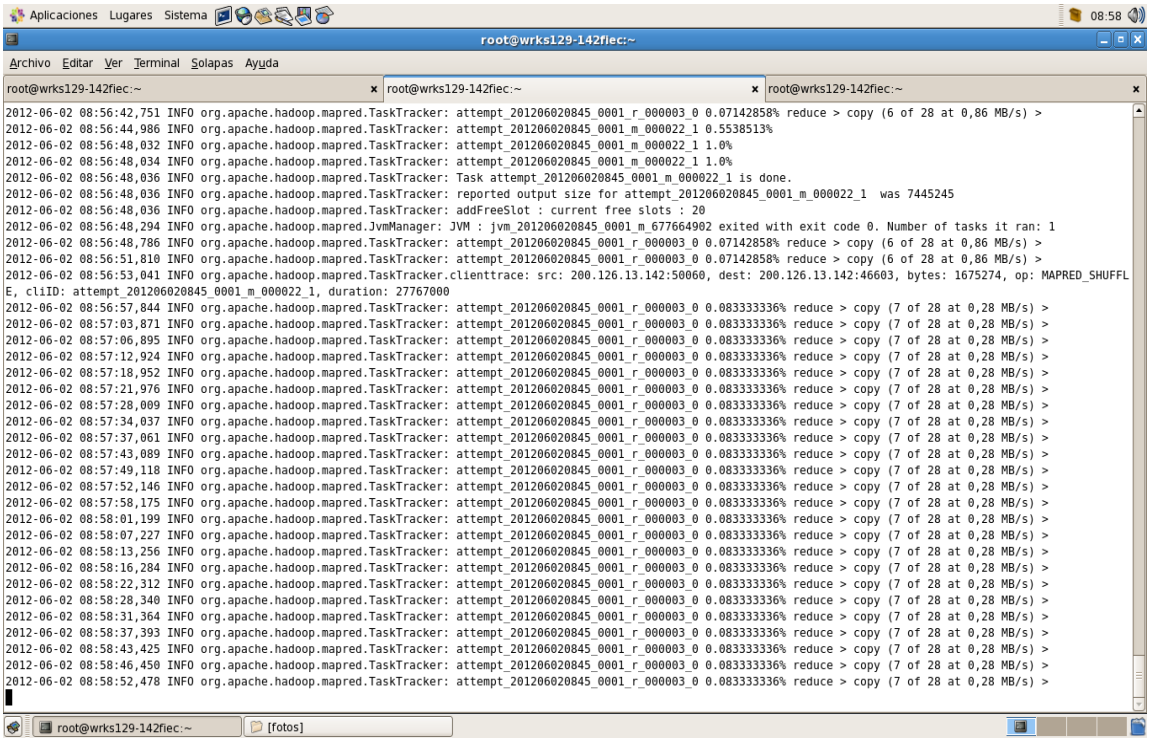

Aplicaciones Lugares Sistema
08:33
root@wrks129-136fiec:~
Archivo Editar Ver Terminal Solapas Ayuda
root@wrks129-136fiec:~
root@wrks129-136fiec:~
2012-06-02 08:33:26,487 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_m_000017_1 1.0%
2012-06-02 08:33:26,488 INFO org.apache.hadoop.mapred.TaskTracker: Task attempt_201206020829_0001_m_000017_1 is done.
2012-06-02 08:33:26,489 INFO org.apache.hadoop.mapred.TaskTracker: reported output size for attempt_201206020829_0001_m_000017_1 was 33026667
2012-06-02 08:33:26,489 INFO org.apache.hadoop.mapred.TaskTracker: addFreeSlot : current free slots : 20
2012-06-02 08:33:26,508 INFO org.apache.hadoop.mapred.JvmManager: JVM : jvm_201206020829_0001_m_1775169987 exited with exit code 0. Number of tasks it ran: 1
2012-06-02 08:33:26,718 INFO org.apache.hadoop.mapred.JvmManager: JVM : jvm_201206020829_0001_m_-1694475091 exited with exit code 0. Number of tasks it ran: 1
2012-06-02 08:33:26,724 INFO org.apache.hadoop.mapred.JvmManager: JVM : jvm_201206020829_0001_m_168851463 exited with exit code 0. Number of tasks it ran: 1
2012-06-02 08:33:27,193 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_r_000003_0 0.047619052% reduce > copy (4 of 28 at 0,25 MB/s) >
2012-06-02 08:33:28,649 INFO org.apache.hadoop.mapred.TaskTracker: LaunchTaskAction (registerTask): attempt_201206020829_0001_m_000018_1 task's state:UNASSIGNED
2012-06-02 08:33:28,649 INFO org.apache.hadoop.mapred.TaskTracker: Trying to launch : attempt_201206020829_0001_m_000018_1 which needs 1 slots
2012-06-02 08:33:28,649 INFO org.apache.hadoop.mapred.TaskTracker: In TaskLauncher, current free slots : 20 and trying to launch attempt_201206020829_0001_m_000018_1 w
hich needs 1 slots
2012-06-02 08:33:28,656 INFO org.apache.hadoop.mapred.JvmManager: In JvmRunner constructed JVM ID: jvm_201206020829_0001_m_1171707767
2012-06-02 08:33:28,656 INFO org.apache.hadoop.mapred.JvmManager: JVM Runner jvm_201206020829_0001_m_1171707767 spawned.
2012-06-02 08:33:28,659 INFO org.apache.hadoop.mapred.TaskController: Writing commands to /root/Desktop/hadoop-1.0.1/mapred_local_dir/ttprivate/taskTracker/root/jobcac
he/job_201206020829_0001/attempt_201206020829_0001_m_000018_1/taskjvm.sh
2012-06-02 08:33:29,302 INFO org.apache.hadoop.mapred.TaskTracker: JVM with ID: jvm_201206020829_0001_m_1171707767 given task: attempt_201206020829_0001_m_000018_1
2012-06-02 08:33:33,225 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_r_000003_0 0.047619052% reduce > copy (4 of 28 at 0,25 MB/s) >
2012-06-02 08:33:34,299 INFO org.apache.hadoop.mapred.TaskTracker.clienttrace: src: 200.126.13.136:50060, dest: 200.126.13.136:59885, bytes: 14804708, op: MAPRED_SHUFF
LE, cliID: attempt_201206020829_0001_m_000025_0, duration: 116421000
2012-06-02 08:33:34,422 INFO org.apache.hadoop.mapred.TaskTracker.clienttrace: src: 200.126.13.136:50060, dest: 200.126.13.136:59885, bytes: 7596027, op: MAPRED_SHUFFL
E, cliID: attempt_201206020829_0001_m_000017_1, duration: 119139000
2012-06-02 08:33:34,460 INFO org.apache.hadoop.mapred.TaskTracker.clienttrace: src: 200.126.13.136:50060, dest: 200.126.13.136:59885, bytes: 4071165, op: MAPRED_SHUFFL
E, cliID: attempt_201206020829_0001_m_000009_1, duration: 33362000
2012-06-02 08:33:35,901 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_m_000018_1 0.010524854%
2012-06-02 08:33:38,922 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_m_000018_1 0.038897067%
2012-06-02 08:33:39,251 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_r_000003_0 0.083333336% reduce > copy (7 of 28 at 0,53 MB/s) >
2012-06-02 08:33:41,949 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_m_000018_1 0.042051926%
2012-06-02 08:33:44,968 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_m_000018_1 0.045486465%
2012-06-02 08:33:45,276 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_r_000003_0 0.083333336% reduce > copy (7 of 28 at 0,53 MB/s) >
2012-06-02 08:33:47,988 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_m_000018_1 0.06921181%
2012-06-02 08:33:48,304 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_r_000003_0 0.083333336% reduce > copy (7 of 28 at 0,53 MB/s) >
2012-06-02 08:33:51,008 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_m_000018_1 0.13673954%
2012-06-02 08:33:54,038 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_m_000018_1 0.14172842%
2012-06-02 08:33:54,330 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_r_000003_0 0.083333336% reduce > copy (7 of 28 at 0,53 MB/s) >
2012-06-02 08:33:57,131 INFO org.apache.hadoop.mapred.TaskTracker: attempt_201206020829_0001_m_000018_1 0.18536814%
root@wrks129-136fiec:~

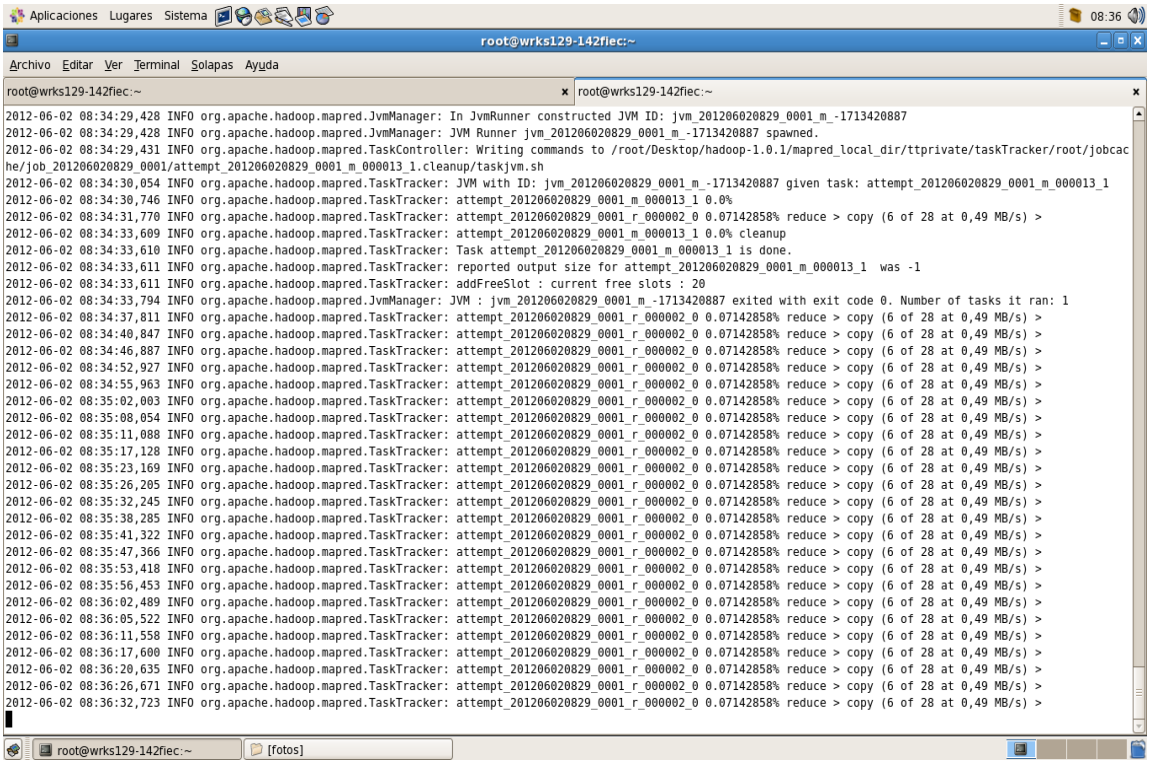
wrks129-137fiec





wrks129-142fiec





BIBLIOGRAFÍA

[1] Kaspersky Lab, “Evolución del spam”, Disponible en línea en: http://www.viruslist.com/sp/spam/info?chapter=153350530, Fecha del último acceso: Octubre del 2011.

[2] Hermann Juergen, “Problema del Spam”, Disponible en línea en: http://www.cauce.org.ar/ProblemaDelSpam, Fecha del último acceso: Octubre del 2011.

[3] Chuck Lam, “Hadoop in Action”, Manning Publications, 2011, 325 Páginas.

[4] Apache Software Foundation, “Apache Hadoop”, Disponible en: http://hadoop.apache.org/ Fecha del último acceso: Noviembre del 2011.

[5] Apache Software Foundation, “HDFS Architecture”, Disponible en línea en: http://hadoop.apache.org/common/docs/r0.20.203.0/hdfs_design.html, Fecha del último acceso: Noviembre del 2011.

[6] Apache Software Foundation, “Single Node Setup”, Disponible en: http://hadoop.apache.org/common/docs/r0.20.203.0/single_node_setup.html, Fecha del último acceso: Noviembre del 2011.

[7] Apache Software Foundation, “Clúster Setup”, Disponible en: http://hadoop.apache.org/common/docs/r0.20.203.0/cluster_setup.html, Fecha del último acceso: Octubre del 2011.

[8] Apache Software Foundation, “MapReduce Tutorial”, Disponible en: http://hadoop.apache.org/common/docs/r0.20.203.0/mapred_tutorial.html, Fecha del último acceso: Noviembre del 2011.

[9] Ghemawat Sanjay – Gobioff Howard – Leung Shun Tak, “The Google File System”, Disponible en línea en: http://research.google.com/archive/gfs.html, Fecha del último acceso: Noviembre del 2011.

[10] Tom White, “Hadoop: The Definitive Guide 2nd Edition”, O’Reilly Media, 2010, 626 Páginas.

[11] G. Dalkilic & M. H. Ozcanhan, (2009). “A simple yet effective spam blocking method. SIN09 Proceddings of the 2nd International Conference on Security of Information and Networks(pp. 179-185)”. Disponible en: http://dx.doi.org/10.1145/1626195.1626241, Fecha del último acceso: Diciembre del 2011.

[12] P. He, X. Wen, & W. Zheng, (2009). “A simple Method for filtering Image Spam.2009 Eight IEEE ACIS International Conference on Computer and Information Science, 910-913 IEEE”. Disponible en: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5223148 Fecha del último acceso: Diciembre del 2011.

[13] J. Dean, & S. Ghemawat, (2008). “MapReduce: Simplified Data Processing onf Large Clusters. Communications of the ACM, 51(1), 1-13, ACM”. Disponible en: http://portal.acm.org/citation.cfm?id=1327492. Fecha del último acceso: Diciembre del 2011.

[14] Apache Software Foundation, “Apache Pig”. Disponible en: http://pig.apache.org/ Fecha del último acceso: Mayo del 2012.

[15] Apache Software Foundation, “Apache Tika”. Disponible en: http://tika.apache.org/ Fecha del último acceso: Mayo del 2012.

[16] Internet FAQ Archives, “RFC 822 – Estándar para el formato de Texto ARPA DE INTERNET”. Disponible en: http://www.faqs.org/rfcs/rfc822.html#b Fecha del último acceso: Mayo del 2012.

[17] Paul Graham, “A Plan for Spam”. Disponible en: http://www.paulgraham.com/spam.html Fecha del último acceso: Mayo del 2012.

[18] Paul Graham, “Better Bayesian Filtering”. Disponible en: http://www.paulgraham.com/better.html Fecha del último acceso: Mayo del 2012.

[19] Paul Graham, “Hackers&Painters First Edition”, O’Reilly Media, 2004, 274 Páginas.