ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

“Generación de recomendaciones de ítems musicales basado en las valoraciones implícitas y las similitudes de los usuarios utilizando Hadoop para procesamientos masivos y escalables”

INFORME DE MATERIA DE GRADUACIÓN

Previo a la Obtención del Título de:

INGENIERO EN COMPUTACIÓN
ESPECIALIZACIÓN EN SISTEMAS DE INFORMACIÓN
INGENIERO EN COMPUTACIÓN
ESPECIALIZACIÓN SISTEMAS TECNOLÓGICOS

Presentada por:

Mervyn Xavier Macías Martrus
Freddy Fernando De la Rosa Amaiquema

GUAYAQUIL – ECUADOR
2009

AGRADECIMIENTO

A nuestras familias, a todos nuestros compañeros de la Materia de Graduación por sus aportaciones y sugerencias que contribuyeron a la realización de nuestro trabajo y de modo muy especial a la Ing. Cristina Abad por compartir sus conocimientos sobre el tema y por darnos todo su apoyo y confianza.

DEDICATORIA

Doy gracias al Señor por haber sido mi guía durante todo este tiempo de carrera profesional, por ser el único camino que me ha llevado hacia la felicidad plena y al encuentro conmigo mismo.

Agradezco a mi familia y a mis padres por sus enseñanzas y consejos en los momentos más difíciles.

A mis hermanos Miguelito y Maru por todo su apoyo; a mi abuelita Rosita Vera por su perseverancia y ayuda incondicional, a Jenifer por todo su cariño y amor y a todos quienes directa e indirectamente tomaron parte en mi formación tanto profesional, personal y espiritual.

Mervyn Macías M.

DEDICATORIA

Dedico este trabajo sobre todo a Dios, mi fortaleza y fuente inagotable de amor.

A mis padres, Fernando y Marcela, que siempre me han demostrado un apoyo y amor incondicional. A mis hermanos, que no dudaron en tenderme la mano cuando lo requerí.

A mi gran amigo, José Luis, y a su familia que me apoyaron siempre.

Freddy De la Rosa.

TRIBUNAL DE GRADO

Ing. Federico Raue R.	Ing. Cristina Abad R.
PROFESOR DELEGADO POR EL DECANO	DIRECTORA DEL CURSO DE GRADUACIÓN

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Proyecto de Graduación, nos corresponden exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

(Reglamento de Graduación de la ESPOL)

Mervyn Xavier Macías Martrus

Freddy Fernando De la Rosa Amaiquema

RESUMEN

El presente trabajo se resume en 5 capítulos. El primer capítulo trata de los sistemas de recomendaciones en general y de varios algoritmos que se utilizan para su implementación.

En el segundo capítulo hacemos un análisis general del proyecto. Aquí se describen las justificaciones de uso de los componentes más relevantes del sistema y se analizan los algoritmos elegidos para la etapa del diseño y posterior implementación.

El tercer capítulo muestra el diseño de los componentes específicos del proyecto. Aquí se podrá observar y entender de modo abstracto el funcionamiento interno de todo el proceso.

El cuarto capítulo presenta la implementación. Aquí se encontrarán pequeños pasos de configuraciones y codificación para llevar a cabo las distintas tareas que el sistema conlleva.

El quinto capítulo muestra los resultados comparativos de dos procesamientos en cuanto al tiempo de respuesta y a los costos incurridos durante el trascurso de todo el proyecto.

INDICE GENERAL

INDICE DE TABLAS

INDICE DE GRÁFICOS

INTRODUCCIÓN

Un sistema de recomendaciones es un tipo específico de filtro de información que ayuda al usuario a seleccionar ítems de su interés tales como películas, músicas, páginas Web, revistas, libros, etc. Actualmente, los sitios Web que prestan estos servicios requieren que la gran cantidad de información recibida por todas las acciones implícitas o explícitas de millones de usuarios sobre millones de ítems, sea procesada de una manera rápida y con la menor infraestructura posible, esto con el fin de obtener rápidos y mejores índices de preferencias útiles y a menor costo.

"Y es que el filtrado colaborativo es un aspecto de gran importancia dentro de las redes sociales y la pequeña evolución que ha supuesto la llamada Web 2.0" [1].

El presente trabajo tiene como objetivo presentar una comparación entre dos procesamientos de recomendaciones de artistas musicales basándose en las preferencias de los usuarios y utilizando un modelo de programación masiva y escalable a partir de un conjunto de datos pre-procesados.

CAPITULO 1

1. SISTEMAS DE RECOMENDACIONES

En la actualidad existen sitios Web que ofrecen una gran cantidad de opciones de selección de productos y servicios, basados en las preferencias de otros usuarios. Entre los más conocidos está Amazon, una compañía estadounidense que ofrece asesoría de forma personalizada en la compra de productos varios a través de la Internet; Facebook[1], preferida por un 62% de la gente adulta mayores de 16 años [2], tiene la capacidad de agrupar a un conjunto de personas con intereses comunes así como la función de búsqueda y sugerencias de amigos; Snooth[3], es una plataforma social para la recomendación de vinos, la cual se basa en valoraciones acumuladas de los usuarios para la obtención de consejos más acertados.

---

[1] Facebook, Available at: http://www.facebook.com/ [Accedido Septiembre 15, 2009]

[2] El caparazon, *"Actitudes, comportamiento, usos, clasificación de los usuarios de las redes sociales"*. Available at: http://www.dreig.eu/caparazon/2009/02/09/actitudes-comportamiento-usos-clasificacion-de-los-usuarios-de-las-redes-sociales/ [Accedido Agosto 26, 2009]

[3] Snooth : Available at: http://www.snooth.com/ [Accedido Septiembre 15, 2009]

Entre los sistemas de recomendaciones más populares basados en ítems musicales se encuentran: iTunes[4], con su motor de recomendación Genius [5], el cual es una aplicación que construye listas de canciones similares para contenidos en iPod; Pandora [6], cuyo inicio fue un proyecto de un grupo de tecnólogos amantes de la música cuyo propósito era ofrecer recomendaciones mediante la clasificación de la música analizando sus "genes", que en realidad son las características que una canción puede tener, como la melodía, armonía, ritmo, instrumentación; y por último LastFm, uno de sitios musicales más populares (casi por los 40 millones de usuarios activos [7]) que utiliza un sistema de recomendaciones llamado Audioscrobbler [8], el cual es un Plugin instalado en las máquinas de los oyentes, y que ofrece a sus usuarios estaciones de radio, cuadros estadísticos, catálogos de artistas y canciones, foros entres otras alternativas correspondientes a la música y sus últimas tendencias.

---

[4] iTunes. Available at: http://www.apple.com/es/itunes/ [Accedido Septiembre 15, 2009]

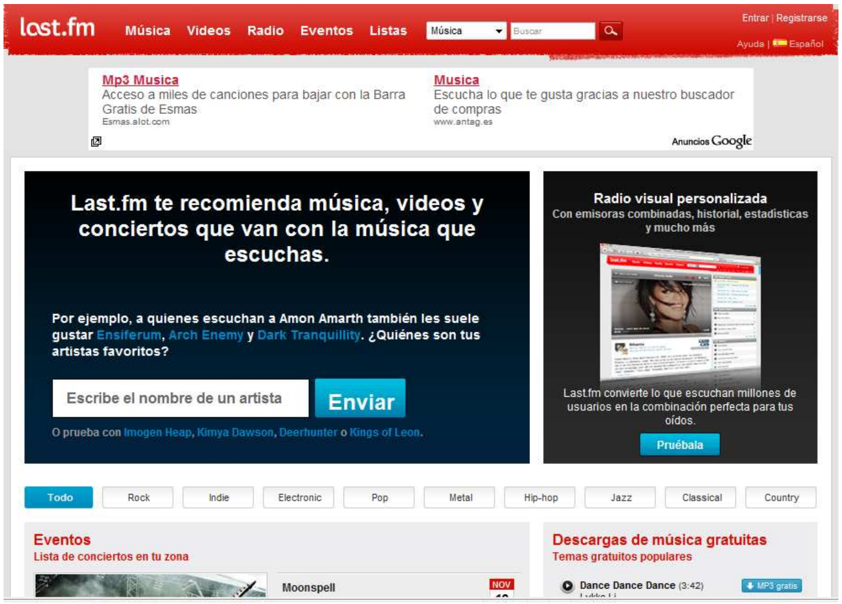
[5] Barra lateral Genius. Available at: http://www.apple.com/es/itunes/whatsnew/ [Accedido Septiembre 6, 2009]

[6] The Music Genome Project. Available at: http://www.pandora.com/mgp.shtml [Accedido Septiembre 6, 2009]

[7] Last.fm estrena su galería de aplicaciones. Available at: http://www.lastfm.es/forum/85167/_/389475 [Accedido Agosto 26, 2009]

[8] Audioscrobbler. The Social Music Technology Playground. Available at: http://www.audioscrobbler.net/ [Accedido Septiembre 6, 2009]

Para utilizar Last.fm es necesario registrarse, luego Last.fm instala el plugin Audioscrobbler que envía al servidor la información correspondiente a los gustos de los usuarios. El usuario debe informar al sistema qué canción le gusta y qué canción no le gusta. El sistema encuentra usuarios con gustos similares para efectuar las recomendaciones.



Mp3 Musica
Acceso a miles de canciones para bajar con la Barra Gratis de Esmas
Esmas.alot.com

Musica
Escucha lo que te gusta gracias a nuestro buscador de compras
www.antag.es

Last.fm te recomienda música, videos y conciertos que van con la música que escuchas.

Por ejemplo, a quienes escuchan a Amon Amarth también les suele gustar Ensiferum, Arch Enemy y Dark Tranquillity. ¿Quiénes son tus artistas favoritos?

O prueba con Imogen Heap, Kimya Dawson, Deerhunter o Kings of Leon.

Radio visual personalizada
Con emisoras combinadas, historial, estadísticas y mucho más

Last.fm convierte lo que escuchan millones de usuarios en la combinación perfecta para tus oídos.

Eventos
Lista de conciertos en tu zona

Descargas de música gratuitas
Temas gratuitos populares
Dance Dance Dance (3:42)

Figura 1.1. Captura de pantalla del sitio Web LastFm

1.1. ¿Qué es un sistema de recomendaciones?

Un sistema de recomendaciones es un tipo específico de filtro de información que ayuda al usuario a seleccionar un ítem de un conjunto de datos sobre los cuales el usuario está interesado. Lo sistemas de recomendaciones juegan el rol de guías que ayudan a los usuarios a tomar decisiones relacionadas con sus gustos. El objetivo de un sistema de recomendaciones es filtrar contenido para proporcionar recomendaciones que satisfagan los requerimientos del usuario.

Es común observar la intervención de los siguientes componentes en un sistema de recomendaciones:

- Usuarios
- Perfiles
- Recomendadores
- Calificación (rating)

Los usuarios son las personas a quienes van dirigidas las recomendaciones, los perfiles modelan al usuario de acuerdo a una variedad de información como el conjunto de ratings que han proporcionado al sistema, los recomendadores son los usuarios que contribuyen a la recomendación y el rating es la calificación que se da a un ítem.

1.2. Mecanismos de retroalimentación

La retroalimentación que recibe el sistema puede ser explícita o implícita [2]. Cuando se aplica la recomendación explícita, el sistema otorga al usuario la oportunidad de calificar, dentro de un rango predefinido, los ítems que ha utilizado. El sistema obtiene retroalimentación implícita capturando la interacción del usuario sin que él lo note. Por ejemplo, en el caso de un sistema de recomendaciones musicales, si un usuario escucha con más frecuencia al artista A que al B, está indicando que prefiere al artista A.

1.3. Métodos de recomendación

En el ámbito más general, existen cinco paradigmas para recomendar al usuario lo que él está buscando [3]:

- Recomendación colaborativa
- Recomendación basada en contenido

- Recomendación basada en demografía
- Recomendación basada en utilidad
- Recomendación basada en conocimiento

En este proyecto nos enfocamos en la recomendación basada en contenido y la recomendación colaborativa. En el primer caso, las recomendaciones son hechas en función de los ítems que un usuario ha elegido en el pasado. En el segundo caso, se identifican un grupo de usuarios con características similares a un usuario, y se recomiendan los ítems que el usuario no haya elegido pero que sean del agrado del grupo de usuarios con los que comparte gustos similares. Actualmente, están emergiendo sistemas que aplican los dos paradigmas, por lo que reciben el nombre de sistemas híbridos, son ejemplos de este tipo Fab [4] y Select [5].

Fab es una implementación distribuida de un sistema híbrido de recomendación de páginas Web. El proceso de recomendación es dividido en dos fases: Colección de ítems para formar un índice, y selección de ítems de ese índice para usuarios particulares.

Select es un sistema de recomendación cuyo objetivo es ayudar a los usuarios de Internet a encontrar de forma rápida y sencilla información confiable, útil, importante y de interés.

1.3.1. Recomendación basada en contenido

Se define la recomendación basada en contenido como aquella en la cual las recomendaciones son hechas exclusivamente en base a los ítems que el usuario ha elegido en el pasado.

1.3.2. Recomendación colaborativa

Se define la recomendación colaborativa como aquella en la cual las recomendaciones son hechas exclusivamente en base a los usuarios con gustos similares. Este método de recomendación no se preocupa del contenido de los ítems, lo único que se conoce acerca de un ítem es un identificador. Utiliza tanto la base de ítems como la base de usuarios para generar predicciones. Emplea producto punto o correlación para encontrar usuarios con características similares, a quienes se denomina vecinos cercanos. Cuando se tiene la lista de vecinos se combina sus preferencias para generar una lista con los N elementos más recomendables para el usuario actual. Los elementos más recomendables son los que tienen mayos valor de recomendación. Para nuestro proyecto elegimos N=5.

1.3.3. Comparación de métodos de recomendación

Cuando se emplea la recomendación basada en contenido se hace un análisis automático de los ítems, considerando atributos predefinidos. Por ejemplo, en un sistema de recomendaciones musicales, se puede considerar el género musical, el país de procedencia del artista, la letra de la canción, etc., pero se deja de lado otros atributos relevantes como la calidad de sonido y la voz del artista. En la recomendación colaborativa, los usuarios utilizan sus criterios para evaluar los ítems, cubriendo así más características de los ítems que evalúan.

En la recomendación basada en contenido las sugerencias son hechas en función del historial de ítems elegidos por el usuario, por lo tanto el usuario sólo puede recibir recomendaciones que concuerden con su perfil. En recomendación colaborativa es posible que el usuario reciba recomendaciones que no se alineen a su perfil.

Una desventaja de la recomendación colaborativa surge cuando un nuevo ítem se añade al sistema. Como no ha sido evaluado por ningún usuario, no hay forma de recomendarlo.

Otro problema de la recomendación colaborativa ocurre cuando un usuario particular no se identifica con los gustos de ningún otro usuario del sistema, en cuyo caso no es factible hallar vecinos cercanos, y por lo tanto hacer recomendaciones.

La recomendación colaborativa requiere un mínimo de usuarios para elaborar las predicciones, problema que se conoce como ColdStar [6].

Si un usuario ha calificado pocos ítems, es probable que tanto el rendimiento como la cantidad de recomendaciones disminuya sobre la búsqueda de vecinos que tengan puntuados los mismos ítems. Entre los paliativos para estos casos están el realizar los procesos de recomendaciones offline y particionar el conjunto de ítems.

1.3.4. Sistema híbrido

Un sistema híbrido utiliza el filtrado basado en contenido y el colaborativo para sus predicciones. La similitud entre usuarios no se define únicamente en función de los ítems que califiquen sino que también es importante que pertenezcan al mismo segmento demográfico. Este método de recomendación puede ser entendido en dos fases: colección de ítems para la creación de un índice y selección de ítems de ese índice para usuarios particulares.

1.4. Algoritmos de recomendación colaborativa

Los algoritmos de recomendación colaborativa se basan en el producto punto entre dos vectores y en las fórmulas de correlación [7].

Producto punto

El primer caso consiste en la aplicación de la expresión con la que se definen el producto punto entre dos vectores.

$$\vec{u} \bullet \vec{v} = |\vec{u}||\vec{v}|\cos\theta$$

La figura 1.2 muestra dos vectores, $\vec{u}$ y $\vec{v}$, donde θ es el ángulo que forman los vectores. Cuando θ es cero, $\vec{u}$ y $\vec{v}$ apuntan en la misma dirección. Así, para valores de θ cercanos a cero, $\vec{u}$ y $\vec{v}$ tienden a apuntar en la misma dirección.

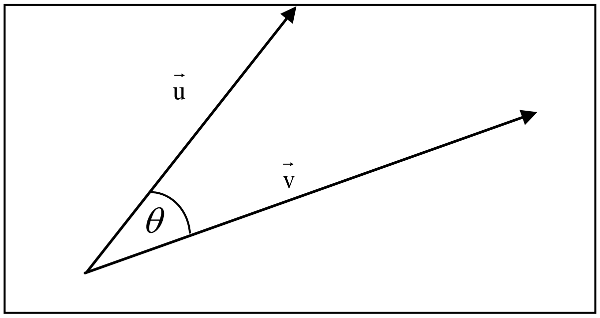


Figura 1.2. Gráfico de dos vectores en el espacio.

Para implementar nuestro proyecto, cada usuario se modela como un vector de n coordenadas, siendo n el número de ítems del sistema. Las coordenadas corresponden a la valoración implícita que el usuario asignó a cada ítem del dataset.

La tabla I muestra un conjunto de datos con 4 usuarios, 5 ítems y el número de veces que cada usuario ha utilizado cada ítem. Por ejemplo, el usuario 3 ha utilizado 4 veces al ítem C.

	Ítems				
Usuario	**A**	**B**	**C**	**D**	**F**
1	4	5	0	0	0
2	5	6	0	5	0
3	5	0	4	0	8
4	0	0	5	0	0

Tabla I. Una base que muestra el número de veces que cada usuario ha escuchado cada ítem

Si queremos calcular qué tan similares son cada par de usuarios del sistema, definimos el vector $\vec{u}_i$ como uno cuyas coordenadas corresponden al número de veces que el usuario *i* escuchó cada ítem. Así por ejemplo, para la tabla I, el vector $\vec{u}_1$ está dado por <4, 5, 0, 0, 0>.

El hecho que haya más similitudes entre los usuarios 1 y 2 que entre los usuarios 1 y 3 indica que el usuario 1 es más similar al usuario 2 que al usuario 3. Esta conclusión se puede obtener directamente calculando el producto punto entre los vectores:

$$\overrightarrow{u_1} \bullet \overrightarrow{u_2} = 50$$

$$\overrightarrow{u_1} \bullet \overrightarrow{u_3} = 20$$

Si para un dataset definimos cada objeto de interés del mismo tipo como un vector, el grado de similitud entre dos objetos cualesquiera es proporcional al coseno del ángulo que forman los vectores, puesto que, mientras menor es el ángulo, mayor es el valor del coseno.

Correlación

El coeficiente de correlación de **Pearson** mide la relación lineal entre dos variables cuantitativas. En este caso las variables cuantitativas son dos usuarios cualesquiera y los valores que toman son las calificaciones implícitas a cada ítem del dataset. Si u y v son dos usuarios, la similitud entre u y v se obtiene a través de la expresión:

$$\mathrm{sim}(u,v) = \frac{\sum_{i=1}^{m}\left(r_{u,i} - r_{\bar{u}}\right)\left(r_{v,i} - r_{\bar{v}}\right)}{\sigma_u \sigma_v}$$

Donde $\mathrm{r}_{\mathrm{u,i}}$ es la calificación que el usuario u asignó al ítem i, $\mathrm{r}_{\bar{\mathrm{u}}}$ es la valoración media del usuario u y σ_{u} es la desviación estándar de las valoraciones del usuario u.

1.4.1. Algoritmos de recomendación basados en usuario

La similitud entre el usuario *i* y el usuario *j* viene dada por la siguiente expresión:

$$\mathrm{Sim}\left(\overrightarrow{\mathrm{u_i}}, \overrightarrow{\mathrm{u_j}}\right) = \frac{\mathrm{u_i} \bullet \mathrm{u_j}}{|\mathrm{u_i}||\mathrm{u_j}|}, \quad 0 \leq \mathrm{Sim}\left(\overrightarrow{\mathrm{u_i}}, \overrightarrow{\mathrm{u_j}}\right) \leq 1$$

Con la aplicación de la fórmula anterior para cada par de usuarios del sistema se genera una matriz de similitud. Por ejemplo, si la primera fila de la matriz de similitud contiene la siguiente información:

u1: 0.5 u2 | 0.3 u3|0.2 u4|0.6 u7|0.9 u8|0.8 u9|0.1 u10

Podemos concluir que el usuario más similar al usuario 1 es el usuario 8.

Para determinar qué usuarios son más similares a un usuario particular se debe seleccionar un umbral. Los usuarios obtenidos se denominan vecinos cercanos. Ahora cada usuario cuenta con un conjunto de vecinos que tienen gustos similares a él. En nuestro proyecto escogimos 5 como umbral. Así, cada usuario tendrá 5 vecinos cercanos.

El sistema debe recomendar ítems que el usuario no haya escuchado, y que sus vecinos cercanos sí lo hayan hecho. Si *n* es el número de vecinos cercanos, *i* es un ítem que no ha sido escuchado por el usuario $\vec{a}$ y $r_{u,i}$ es la valoración que el usuario $\vec{u}$ ha asignado al ítem *i*, el grado de recomendación del ítem *i* al usuario $\vec{a}$ viene dado por la expresión:

$$p_{a,i} = \frac{\sum_{k=1}^{n} r_{u_k,i} \bullet \mathrm{Sim}\left(\vec{a}, \overrightarrow{u_k}\right)}{\sum_{k=1}^{n} \mathrm{Sim}\left(\vec{a}, \overrightarrow{u_k}\right)}$$

1.4.2. Algoritmos de recomendación basados en ítems

El principio es el mismo del apartado anterior, la diferencia es que en este caso buscamos similitudes entre ítems en lugar de buscar similitudes entre usuarios. En nuestro proyecto no implementaremos este algoritmo debido a que es ineficiente como veremos en el apartado siguiente.

1.4.3. Análisis de Algoritmos

Con respecto a la tabla I, al aplicar el algoritmo de recomendación basado en usuario, cada usuario se compara con el resto, así, para cada usuario se realizan 3 comparaciones. Por lo tanto es necesario un total de 12 comparaciones. En el peor de los casos, si cada usuario ha calificado todos los ítems del sistema, cada comparación implica 5 operaciones, por consiguiente el tiempo de ejecución es del orden 12*5=60.

En general, para un dataset con n usuarios y m ítems, para cada usuario se deben realizar n-1 comparaciones, en total n(n-1). En el peor de los casos cada comparación implica m operaciones. Así, el tiempo de ejecución es del orden de mn^2.

El dataset utilizado en este proyecto tiene aproximadamente 150.000 usuarios y 2’000.000 de ítems. El tiempo de ejecución es del orden de 10^{16}.

El resultado anterior es válido tanto para el algoritmo de **Pearson** como para el producto punto. Sin embargo, para el primer caso, el cálculo es más costoso debido a la serie de productos en el numerador y denominador.

CAPITULO 2

2. ANALISIS DEL PROYECTO

2.1. Objetivo

El objetivo es implementar un sistema de recomendaciones musicales aplicando dos algoritmos de recomendación para luego compararlos, basándonos en las preferencias de los usuarios, utilizando un modelo de programación masiva y escalable a partir de un conjunto de datos pre-procesados.

2.2. Alcance

Cada usuario del sistema recibirá 5 recomendaciones de artistas que no haya escuchado en el pasado. Escogimos ese valor porque estimamos que una cantidad mayor incrementaría el tiempo de ejecución en forma considerable y una cantidad menor no dejaría satisfecho al usuario.

2.3. Datasets

Un dataset es una colección de datos presentados, por lo general, en forma tabular, de tal manera que cada columna representa una variable particular y cada fila corresponde a un miembro del dataset. La fila contiene los valores de las variables correspondientes a cada columna. Cada valor recibe el nombre de dato. Existen varios datasets disponibles de forma gratuita en Internet. A continuación nos referiremos a dos de ellos.

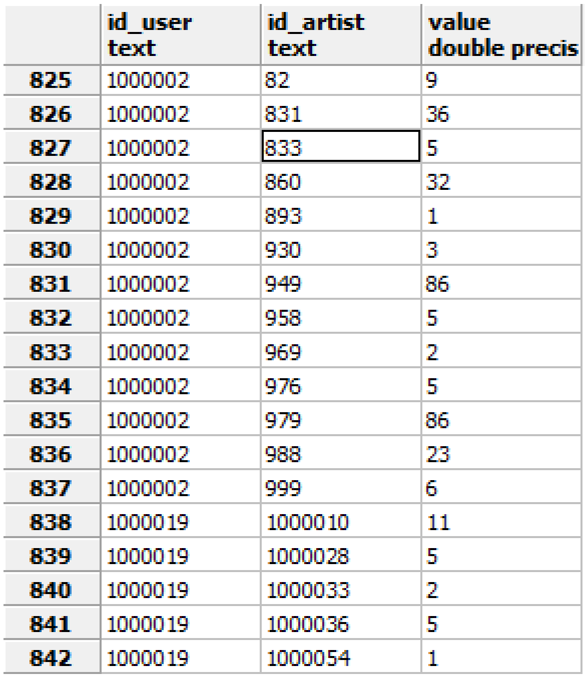
MusicBrainz[9] es un dataset de música mantenido por la comunidad de usuarios. Contiene datos como el nombre del artista, título del álbum lanzado y la lista de pistas que aparecen en un álbum lanzado.

Audioscrobbler[10] es un plugin que se instala en un programa cliente. Cuando un usuario escucha una canción el plugin envía información que actualiza el dataset. El dataset de Audioscrobbler contiene una columna de usuarios, una de artistas y el número de veces que cada usuario escuchó a cada artista, ver figura 2.1.

---

[9] MusicBrainz, Database. Available at: http://musicbrainz.org/doc/Database [Accedido Septiembre 6, 2009]

[10] Audioscrobbler, which is now merged with last.fm. Available at: http://www-etud.iro.umontreal.ca/~bergstrj/audioscrobbler_data.html [Accedido Septiembre 6, 2009]



	id_user text	id_artist text	value double precis
825	1000002	82	9
826	1000002	831	36
827	1000002	833	5
828	1000002	860	32
829	1000002	893	1
830	1000002	930	3
831	1000002	949	86
832	1000002	958	5
833	1000002	969	2
834	1000002	976	5
835	1000002	979	86
836	1000002	988	23
837	1000002	999	6
838	1000019	1000010	11
839	1000019	1000028	5
840	1000019	1000033	2
841	1000019	1000036	5
842	1000019	1000054	1

Figura 2.1. Parte del dataset de Audioscrobbler. La primera columna contiene el Id de usuario; la segunda, el Id del artista; y la tercera, el número de veces que el usuario escuchó al artista.

2.3.1. Selección del dataset

Para la implementación de nuestro proyecto hemos utilizado el dataset de AudioScrobbler como base real de datos acerca de ítems musicales. La base no recibe recomendaciones explícitas de los usuarios. Utilizaremos la siguiente aproximación: Si un usuario ha escuchado un ítem A más veces que un ítem B, A es más agradable que B para él.

Audioscrobbler es un dataset de uso extendido, contiene recomendaciones implícitas de los usuarios y no tiene información relevante de contenido, por lo que es ideal para aplicar el método de recomendación colaborativa.

El dataset cuenta aproximadamente con 150.000 usuarios, 2’000.000 de artistas y 24’000.000 de recomendaciones implícitas. El archivo tiene los siguientes campos: un identificador del usuario, un identificador del ítem y una valoración que ese usuario ha asignado a ese ítem. Cabe recalcar que la valoración es el número de instancia del ítem que el usuario ha solicitado o utilizado, por lo tanto se trata de una valoración implícita. No se utilizó valoración explícita debido a que estos presentan varios problemas como el uso apropiado de escalas, motivación e incentivo para los evaluadores, alcance de cantidad crítica de usuarios, credibilidad en las votaciones, imposición del costo cognitivo para el evaluador, etc [5].

2.4. Selección del Método

Para la implementación elegimos el método de recomendación colaborativa basada en usuarios.

A más de las ventajas de la recomendación colaborativa con respecto a la recomendación basada en contenido que se analizaron en el capítulo anterior, elegimos la recomendación colaborativa porque los datasets disponibles no tienen información relevante sobre el contenido de los ítems. En el caso de AudioScrobbler lo único que conocemos acerca de un ítem es un identificador.

Elegimos la recomendación colaborativa basada en usuario en lugar de recomendación colaborativa basada en ítem por eficiencia. La recomendación basada en usuario implica 150.000*150.000 comparaciones, mientras que para recomendación basada en ítem se necesitan 2’000.000*2’000.000 comparaciones.

2.5. Selección de algoritmos

Para calcular los vecinos cercanos utilizamos los algoritmos que analizamos en el capítulo 1: el producto punto y el coeficiente de correlación de **Pearson**.

Escogimos el algoritmo del producto punto para ahorrar tiempo de ejecución. Y escogimos el algoritmo del coeficiente de correlación de **Pearson** porque hay una mayor fidelidad en los resultados.

2.6. Herramientas

A continuación se describe brevemente el modelo de programación y las tecnologías sobre la cual se implementaron los algoritmos de recomendación.

2.6.1. Paradigma map-reduce

Map-reduce es un modelo de programación desarrollado por Google para soportar programación en paralelo sobre grandes cantidades de datos [8]. El programador debe implementar dos funciones **map** y **reduce**. La función **map** toma una lista de pares y retorna un conjunto de pares clave/valor. El framework agrupa los valores intermedios asociados con la misma clave y los envía a la función **reduce**. La función toma una clave intermedia y un conjunto de valores para esa clave, produciendo, por lo general, una o ninguna salida.

En este proyecto utilizamos el paradigma map-reduce por la necesidad de implementar en paralelo los algoritmos de recomendación debido a la cantidad de elementos del dataset. En el capítulo 3 se analiza con más detalle este paradigma de programación.

2.6.2. Framework Hadoop

Hadoop es un proyecto de Lucene que implementa las tecnologías de Google map-reduce y Google File System [9]. El equivalente al Google File System, es el Sistema de Archivos Distribuidos Hadoop (HDFS por sus siglas en inglés), que es un sistema de archivos tolerante a fallos, útil para aplicaciones que manejan grandes cantidades de datos. Hadoop permite distribuir los datos entre nodos, por lo tanto es factible el trabajo en paralelo. Si bien nuestro proyecto no cuenta con archivos en el orden de gigabytes, la cantidad de usuarios e ítems requiere un alto grado de paralelismo por el número de cálculos inmersos en la implementación de los algoritmos de recomendación colaborativa.

2.6.3. Librería Mahout

Mahout es un proyecto de Apache Lucene[11] desarrollado en Java cuyo propósito es construir librerías escalables de máquinas de aprendizaje. Mahout implementa el algoritmo basado en ítems y el algoritmo basado en contenido [10].

Para trabajar con los algoritmos de Mahout se necesita una gran cantidad de memoria RAM debido a que el modelo de datos se implementa mediante la interfaz DataModel. Mahout crea un objeto usuario por cada usuario del sistema y recomienda un dataset con un máximo de diez mil usuarios. Con estas limitaciones no es factible implementar nuestro proyecto con esta librería. Nos vemos en la necesidad de manipular el código fuente para utilizar el dataset completo.

---

[11] Apache Mahout - Overview . Available at: http://lucene.apache.org/mahout/ [Accedido Septiembre 6, 2009]

2.6.4. Amazon Web Services - EC2

Amazon EC2 (Elastic Compute Cloud)[12] es un servicio de Amazon que permite a los usuarios alquilar plazos de tiempo para ejecutar máquinas virtuales en la nube. Está diseñado para facilitar la programación escalable a los desarrolladores. Un cliente de Amazon EC2 puede crear, correr y terminar instancias cuando lo necesite.

2.6.5. Amazon Web Services - S3

Amazon S3[13] es un servicio de Amazon que permite el almacenamiento por Internet. Al igual que EC2, está diseñado para facilitar la programación en paralelo a los desarrolladores. Amazon S3 ofrece una simple interfaz Web que puede ser utilizada para almacenar y recuperar datos. Entre las ventajas que ofrece se encuentran, escalabilidad, confiabilidad, rapidez y bajo costo. Para ejecutar el proyecto cargamos el dataset de Audiosrobbler en Amazon S3.

---

[12] Amazon Elastic Computed cloud (Amazon EC2). Available at: http://aws.amazon.com/ec2/ [Accedido Septiembre 6, 2009]

[13] Amazon Simple Storage Service (Amazon ES2). Available at: http://aws.amazon.com/s3/ [Accedido Septiembre 6, 2009]

CAPITULO 3

3. Diseño del Proyecto

3.1. Arquitectura del Sistema

La infraestructura necesaria para correr nuestro proceso requiere de un equipo operador con distribución para Windows, un sistema de virtualización por software y una cuenta registrada de Amazon para utilizar sus servicios Web.

Los requerimientos del equipo operador son los siguientes:

1. Características del equipo:
    a. Tener un mínimo de 2 GB de memoria.
    b. Tener un disco duro con un mínimo de 250 GB.
    c. Procesador Core 2 Duo.
    d. Sistema operativo Windows (versión XP o superior)
    e. Conexión a internet con un ancho de banda mínimo de 500kbps.
2. Software Instalado
    a. Un cliente SSH, Telnet con licencia libre (Putty versión 0.6)

b. Un ambiente de desarrollo (Eclipse en su versión 3.4.2)
c. Java SDK (versión 1.4.2 o superior).
d. Un navegador Web multiplataforma (Firefox)
e. Un software de virtualización (Vmware-Workstation versión 6.5.1)
f. Un cliente FTP con licencia libre (Filezilla versión 3.1.2)

Navegador Web multiplataforma

El navegador Web elegido para nuestro proyecto fue Firefox en su versión 5.0. Además de ser de distribución libre, este navegador posee plugins que ayudan en la organización de los archivos en el almacén de Amazon S3 y en el monitoreo de ejecución de los procesos en EC2.

Para utilizar los dos componentes anteriormente mencionados, se listan los siguientes pre-requisitos:

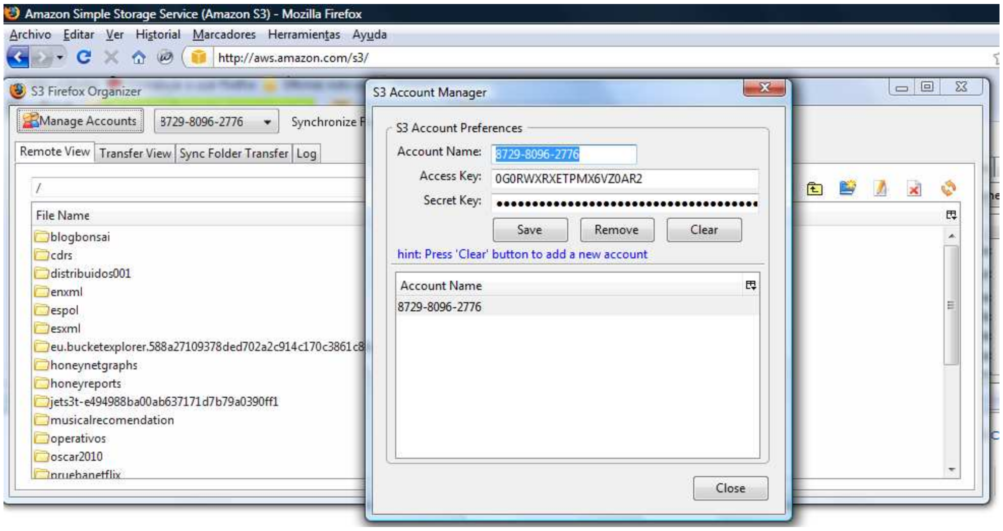
- Una cuenta para utilizar los servicios Web de Amazon S3 (AWS en sus siglas en inglés), el cual contiene un código de acceso y un código de acceso secreto.

- Es recomendable un ancho de banda de por lo menos 500 kbps.

Las acciones de petición y transferencia de archivos desde y hacia el S3 conllevan a cargas sobre la cuenta adquirida. En el capítulo 5 de la sección de resultados se muestran las tablas de precios y los cálculos para la estimación de costos incurridos.

Firefox plugin: Organizador de archivos en S3

El plugin "Amazon S3 Firefox Organizer (S3Fox) 0.4.7" nos permite la organización de nuestros archivos en un bucket y que luego serán utilizados en EC2. Además permite otorgar permisos de lectura, escritura y ejecución sobre los archivos mediante las políticas de control de acceso (ACL en sus siglas en inglés). La figura 3.1 muestra la interfaz del organizador de archivos de firefox en Amazon S3.



Account Name: 8729-8096-2776

Access Key: 0G0RWXRXETPMX6VZ0AR2

Secret Key: ••••••••••••••••••••••••••••••••••••••

hint: Press 'Clear' button to add a new account

Account Name
8729-8096-2776

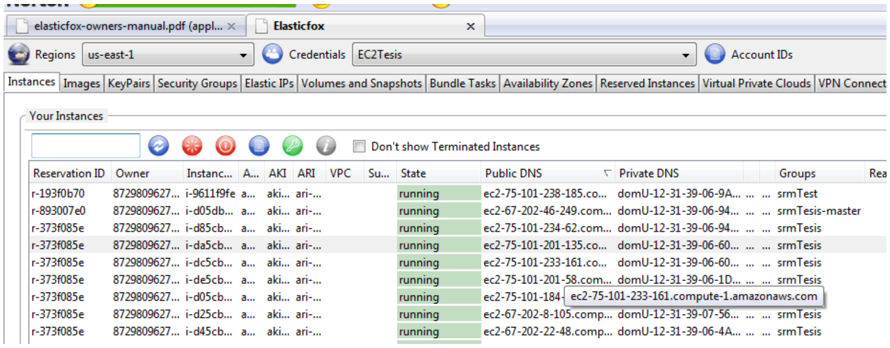
/

File Name
blogbonsai
cdrs
distribuidos001
enxml
espol
esxml
eu.bucketexplorer.588a27109378ded702a2c914c170c3861c8
honeynetgraphs
honeyreports
jets3t-e494988ba00ab637171d7b79a0390ff1
musicalrecomendation
operativos
oscar2010
pruebanetflix

Figura 3.1. Captura de pantalla del Amazon S3 Firefox Organizer (S3Fox).

Firefox plugin: Monitor de instancias en EC2

El plugin Elasticfox para firefox es una extensión para Amazon EC2. Este componente permite el monitoreo de las instancias (nodos participantes) que están corriendo en EC2; además, administra grupos de seguridad asociados a las instancias levantadas. En nuestro caso esta interfaz nos ayudó con el monitoreo del estado de ejecución del proceso dentro del grupo de seguridad creado “srmTesis” en el cluster. Además nos permitió otorgar los permisos sobre el protocolo http en el puerto 50030 (para la interfaz Web del JobTracker permitiendo ver el estado del proceso) y 50060 (para la interfaz Web del TaskTracker para ver detalles del debug) necesarios para las máquinas fuera de este grupo de seguridad mencionado. A continuación las figuras 3.2 y 3.3 muestran la interfaces correspondientes al monitoreo de instancias y al mantenimiento de los permisos de seguridad de grupos en Amazon EC2 respectivamente.



Regions: us-east-1

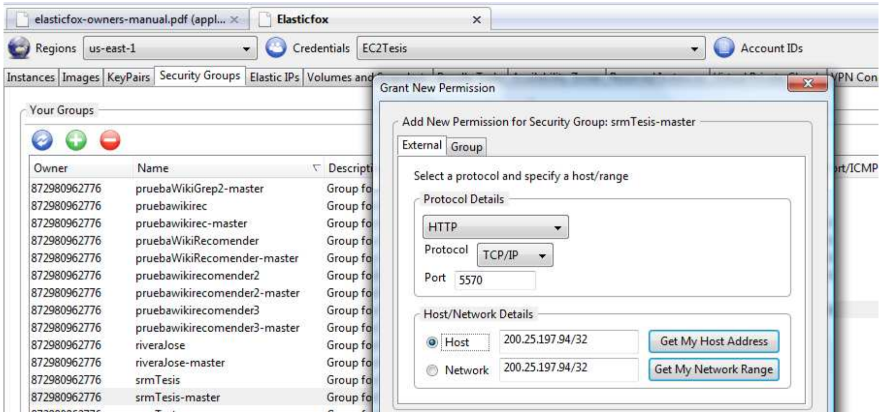
Credentials: EC2Tesis

☐ Don't show Terminated Instances

Reservation ID	Owner	Instanc...	A...	AKI	ARI	VPC	Su...	State	Public DNS	Private DNS			Groups	Rea
r-193f0b70	8729809627...	i-9611f9fe	a...	aki...	ari-...			running	ec2-75-101-238-185.co...	domU-12-31-39-06-9A...	...	...	srmTest	
r-893007e0	8729809627...	i-d05db...	a...	aki...	ari-...			running	ec2-67-202-46-249.com...	domU-12-31-39-06-94...	...	...	srmTesis-master	
r-373f085e	8729809627...	i-d85cb...	a...	aki...	ari-...			running	ec2-75-101-234-62.com...	domU-12-31-39-06-94...	...	...	srmTesis	
r-373f085e	8729809627...	i-da5cb...	a...	aki...	ari-...			running	ec2-75-101-201-135.co...	domU-12-31-39-06-60...	...	...	srmTesis	
r-373f085e	8729809627...	i-dc5cb...	a...	aki...	ari-...			running	ec2-75-101-233-161.co...	domU-12-31-39-06-60...	...	...	srmTesis	
r-373f085e	8729809627...	i-de5cb...	a...	aki...	ari-...			running	ec2-75-101-201-58.com...	domU-12-31-39-06-1D...	...	...	srmTesis	
r-373f085e	8729809627...	i-d05cb...	a...	aki...	ari-...			running	ec2-75-101-184-					
r-373f085e	8729809627...	i-d25cb...	a...	aki...	ari-...			running	ec2-67-202-8-105.comp...	domU-12-31-39-07-56...	...	...	srmTesis	
r-373f085e	8729809627...	i-d45cb...	a...	aki...	ari-...			running	ec2-67-202-22-48.comp...	domU-12-31-39-06-4A...	...	...	srmTesis	

ec2-75-101-233-161.compute-1.amazonaws.com

Figura 3.2. Captura de pantalla en el momento de ejecución de las recomendaciones simples sobre el grupo "srmTesis"



Regions: us-east-1

Credentials: EC2Tesis

Owner	Name	Descripti
872980962776	pruebaWikiGrep2-master	Group fo
872980962776	pruebawikirec	Group fo
872980962776	pruebawikirec-master	Group fo
872980962776	pruebaWikiRecomender	Group fo
872980962776	pruebaWikiRecomender-master	Group fo
872980962776	pruebawikirecomender2	Group fo
872980962776	pruebawikirecomender2-master	Group fo
872980962776	pruebawikirecomender3	Group fo
872980962776	pruebawikirecomender3-master	Group fo
872980962776	riveraJose	Group fo
872980962776	riveraJose-master	Group fo
872980962776	srmTesis	Group fo
872980962776	srmTesis-master	Group fo

Add New Permission for Security Group: srmTesis-master

Select a protocol and specify a host/range

Protocol Details

HTTP

Protocol: TCP/IP

Port: 5570

Host/Network Details

● Host: 200.25.197.94/32

○ Network: 200.25.197.94/32

Figura 3.3. Captura de pantalla en el momento de otorgar permisos en el puerto 50030 para máquinas fuera del grupo de seguridad.

Software de Virtualización

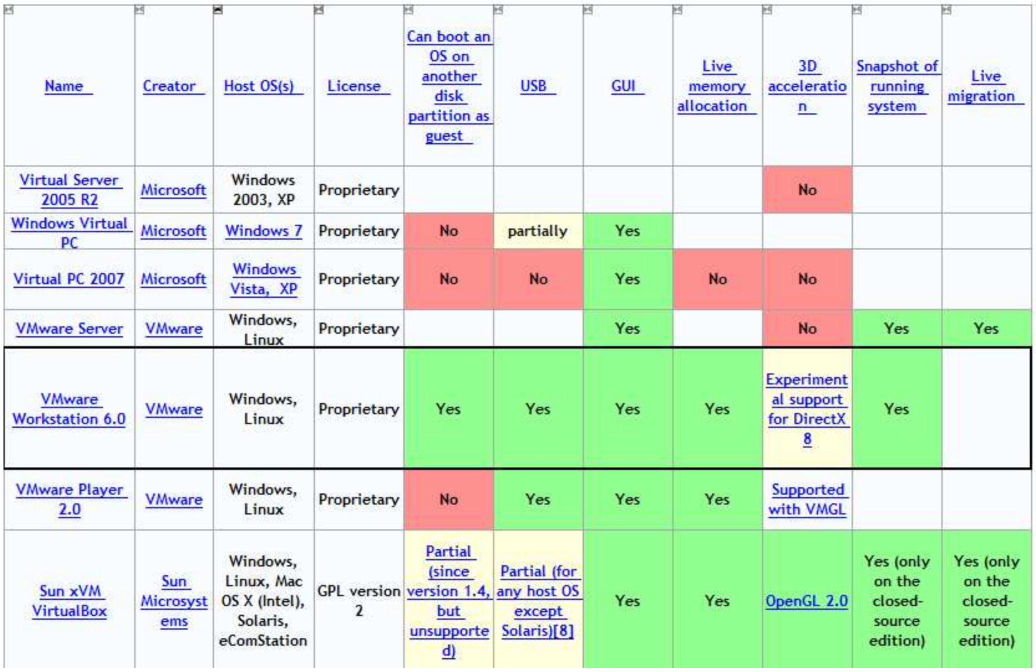
Para el sistema de virtualización se eligió utilizar el software VMware workstation en su versión 6.0. Este software presenta un aprovechamiento óptimo de los recursos permitiendo correr varias instancias de sistemas operativos compatibles con x86 o x86-64. Además posee una característica de restauración del sistema en un punto determinado de estabilidad mediante la herramienta **Snapshot** (foto) el cual permite regresar a un estado guardado en cualquier tiempo.

También posee la ventaja de creación de múltiples máquinas virtuales representando una red local y designando una máquina virtual como administrador, simulando así ambientes de pruebas de dos capas cliente-servidor. Para nuestro caso no se necesitó la creación de múltiples máquinas virtuales ya que únicamente nos bastó tener una máquina virtual con una pre-instalación del framework de Hadoop configurado en modo pseudo-distribuido. Este modo posee los mismos componentes que una distribución completa[14], ideal para el desarrollo y testing.

En la figura 3.4 se muestra un gráfico comparativo que nos sirvió de referencia en la elección del sistema de virtualización.

---

[14] Ubuntu / Debian Packages for Hadoop | Cloudera. *Installing Hadoop (Pseudo-Distributed Mode)*. Available at: http://www.cloudera.com/hadoop-ubuntu#installing_hadoop__pseudodistributed_mode_ [Accedido Septiembre 1, 2009].



Name	Creator	Host OS(s)	License	Can boot an OS on another disk partition as guest	USB	GUI	Live memory allocation	3D acceleration	Snapshot of running system	Live migration
Virtual Server 2005 R2	Microsoft	Windows 2003, XP	Proprietary					No		
Windows Virtual PC	Microsoft	Windows 7	Proprietary	No	partially	Yes				
Virtual PC 2007	Microsoft	Windows Vista, XP	Proprietary	No	No	Yes	No	No		
VMware Server	VMware	Windows, Linux	Proprietary			Yes		No	Yes	Yes
VMware Workstation 6.0	VMware	Windows, Linux	Proprietary	Yes	Yes	Yes	Yes	Experimental support for DirectX 8	Yes	
VMware Player 2.0	VMware	Windows, Linux	Proprietary	No	Yes	Yes	Yes	Supported with VMGL		
Sun xVM VirtualBox	Sun Microsystems	Windows, Linux, Mac OS X (Intel), Solaris, eComStation	GPL version 2	Partial (since version 1.4, but unsupported)	Partial (for any host OS except Solaris)[8]	Yes	Yes	OpenGL 2.0	Yes (only on the closed-source edition)	Yes (only on the closed-source edition)

Figura 3.4. Cuadro de comparación entre soluciones de virtualización para plataformas Windows[15].

Una vez adquirido todo el equipo y software necesario, podemos levantar las instancias necesarias para correr el proceso de generación de recomendaciones. La arquitectura del lado del cluster de Amazon depende de cuántos nodos o instancias se levanten y cómo Amazon tiene estructurado su red interna. De

---

[15] Comparison of platform virtual machines - Wikipedia, the free encyclopedia. Available at: http://en.wikipedia.org/wiki/Comparison_of_platform_virtual_machines [Accedido Septiembre 1, 2009].

nuestro lado sólo fue necesario tener un computador que sirvió como estación de comunicación con una máquina virtual en Linux que almacene la configuración para la conexión con EC2.

La figura 3.5 muestra la infraestructura requerida en un nivel general en donde el “Operador”, a través del VMware con una distribución en Linux (Ubuntu) y pre-instalado el framework de Hadoop, levanta un determinado número de nodos en la nube de EC2; la nube cuenta con (N – 1) “Nodos esclavos” encargados de la ejecución de las tareas y un “Nodo Maestro” encargado de distribuir las tareas y de recibir los resultados que emitan los nodos esclavos. Así los resultados finalmente son almacenados en el S3 de Amazon.

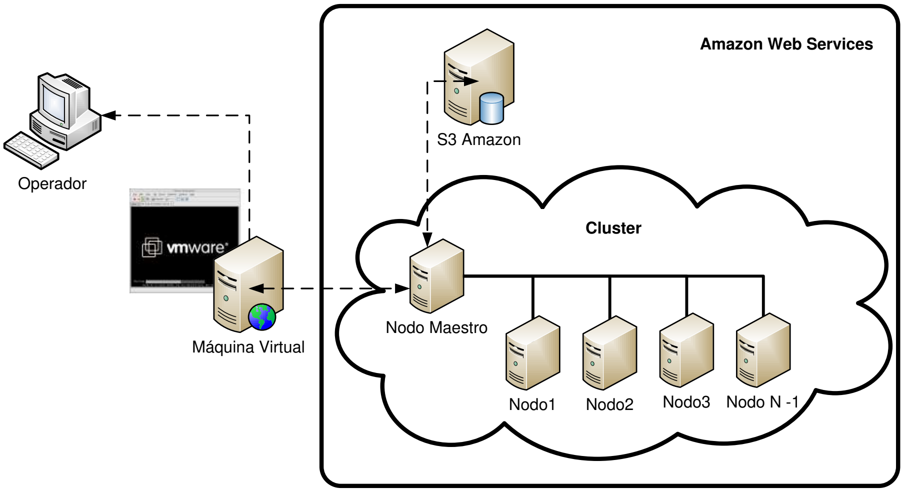


Figura 3.5. Diagrama del requerimiento de la infraestructura tecnológica.

3.2. Diagrama de clases

Para efectos de comparación de rendimiento, se han diseñado dos alternativas para la obtención de las recomendaciones basadas en usuario:

El siguiente diagrama de clases de la figura 3.6 muestra la primera alternativa, basada en el cálculo del producto punto. Este esquema considera medidas de similitud simétrica (si U1 = U2 entonces U2 = U1) el cual provee una solución eficiente a los problemas de escalabilidad de los usuarios [11].

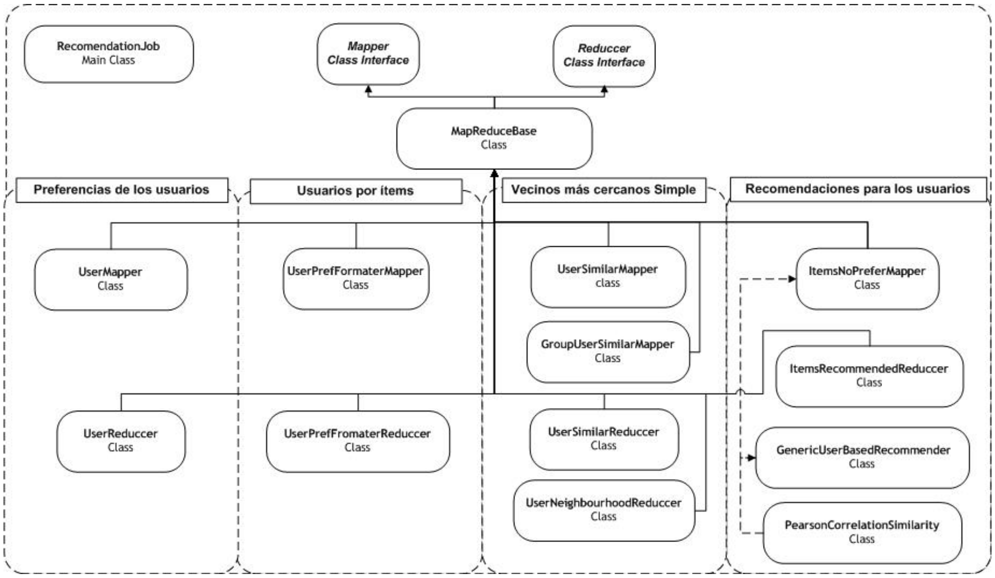


Figura 3.6. Diagrama de clases utilizando producto punto.

En el siguiente diagrama de clases de la figura 3.7 muestra la segunda alternativa basada en la obtención de valores más precisos para la similitud que hay entre un usuario y otro. Estos valores, obtenidos por el algoritmo del coeficiente de correlación de **Pearson** [1], reflejan una cercanía más exacta que está basada en las preferencias que cada uno tiene por sus ítems.

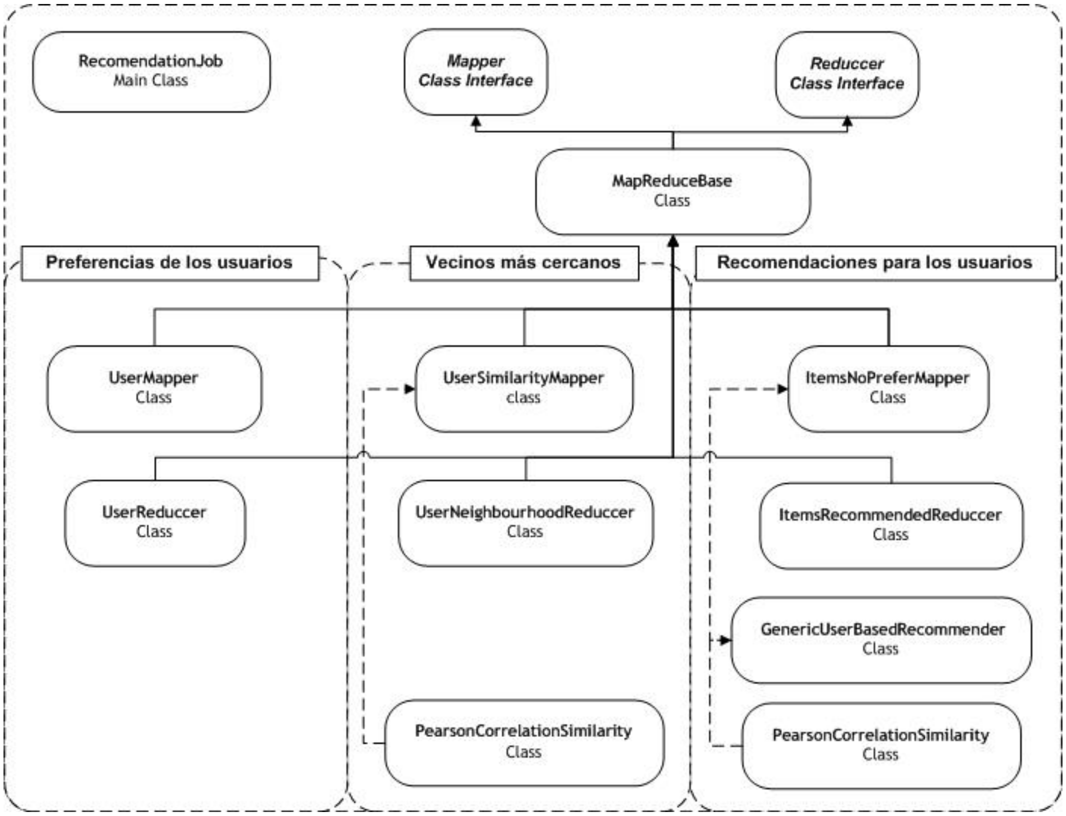


Figura 3.7. Diagrama de clases utilizando una similitud de **Pearson**.

3.3. Esquema map-reduce

Map-reduce es un modelo de programación para el procesamiento de grandes cantidades de información en donde una función llamada **"map"** procesa pares clave/valor para generar un conjunto de pares intermedios clave/valor, y una función **"reduce"** que agrupa todos los valores intermedios asociados con la misma clave [8].

Nuestro trabajo funciona bajo este esquema, el cual se divide en varios módulos que procesan un conjunto clave/valor necesario para la generación de recomendaciones de todos los usuarios.

Al iniciar la función de despliegue de tareas por el nodo maestro, el input, que en nuestro caso es un archivo de texto plano que contiene todas las preferencias de los usuarios, se divide en M pedazos de 16 a 64 MB. El número de pedazos de archivos es el número de tareas **map** en donde cada una es asignada por el nodo maestro a sus nodos trabajadores. Las salidas producidas por los **mappers** estarán en la memoria de cada nodo y periódicamente estas salidas serán escritas en cada disco local cuya localización es enviada al nodo maestro para el posterior trabajo con los **reducers** [8]. Cabe recalcar que en este esquema la ejecución de las tareas **reduce** vendrá luego de que la última tarea **map** haya sido procesada, así los **reducers** leerán, agruparán y procesarán todas las clave/valor generadas por los **mappers**. Al final los **reducers** generarán uno o varios archivos de salida que podrán ser utilizados como entradas para otros procesos bajo el mismo esquema.

La primera etapa del esquema es un pre-procesamiento de las preferencias de cada usuario mediante el archivo de entrada (dataset). En la figura 3.8 se muestra como las tareas **map** envían como clave el id del usuario y como valor una tupla compuesta por el id del ítem y la valoración respectiva dada por el usuario, esto con el fin de que los **reducers** agrupen las claves idénticas y generen archivos de texto plano en donde cada línea este compuesta por todos los artistas preferidos de un usuario con su respectivo valoración separados por un carácter de límite.

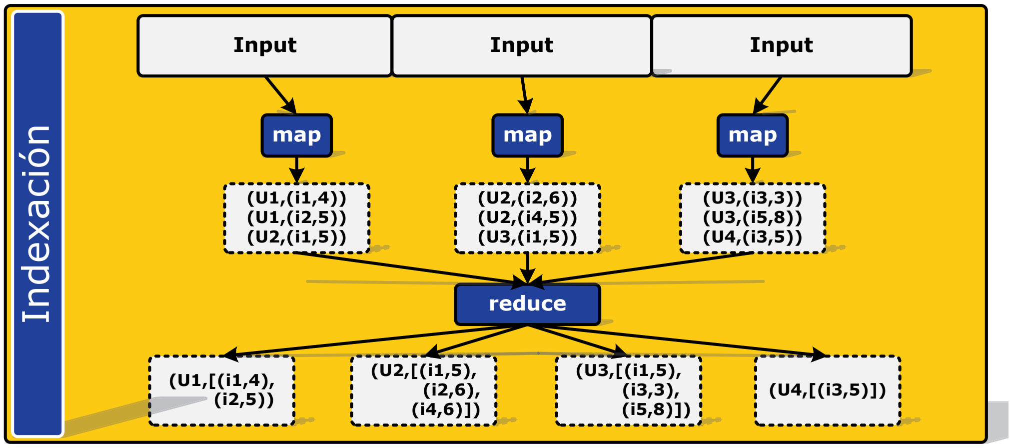


Figura 3.8. Esquema de indexación y pre-procesamiento del dataset

El segundo esquema se trata de la obtención de las similitudes entre usuarios. Este proceso recibe como entrada la salida generada por la primera etapa. Dentro de este esquema y por medio de los algoritmos provistos por la librería de Mahout para la obtención del valor de correlación de **Pearson** y la selección de los vecinos más cercanos, se obtienen valores de similitud para los N usuarios que más se aproximan a las preferencias de cada usuario del conjunto de datos inicial. En la figura 3.9 las tareas **map** emiten como clave/valor el id usuario y una tupla compuesta por un vecino y un valor de correlación respectivamente. Los **reducers** agrupan las claves idénticas formando así un archivo de texto plano donde cada línea está compuesta por los N vecinos más cercanos del usuario acompañado de su valor de correlación.

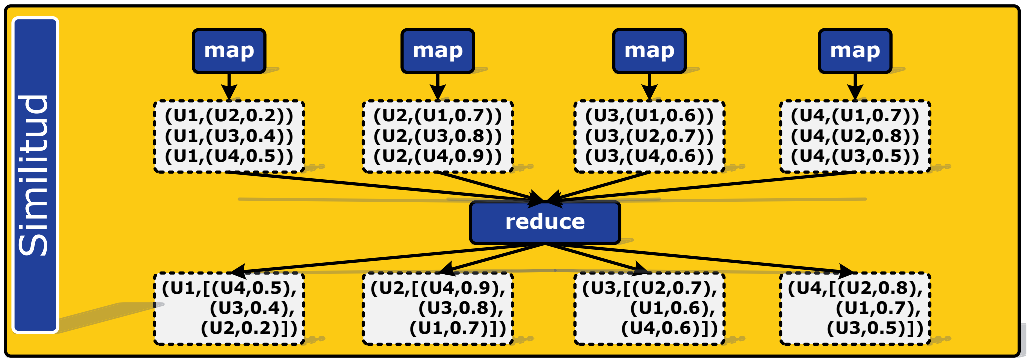


Figura 3.9. Esquema de generación de similitudes entre usuarios

El tercer esquema se trata de la obtención de valores de preferencia estimados para cada uno de los ítems nunca antes vistos por un usuario y así mismo, por medio de la librería Mahout y su algoritmo de recomendaciones basados en usuarios se obtienen los N ítems mejores recomendados para cada usuario. En la figura 3.10 los **mappers** emiten como clave/valor el id del usuario y una tupla compuesta por un ítem nunca antes visto y obtenido de sus vecinos acompañado de un valor estimado preferencial respectivamente. Los **reducers** agrupan las claves idénticas formando así un archivo de texto plano donde cada línea está compuesta por los N ítems mejores recomendados para el usuario.

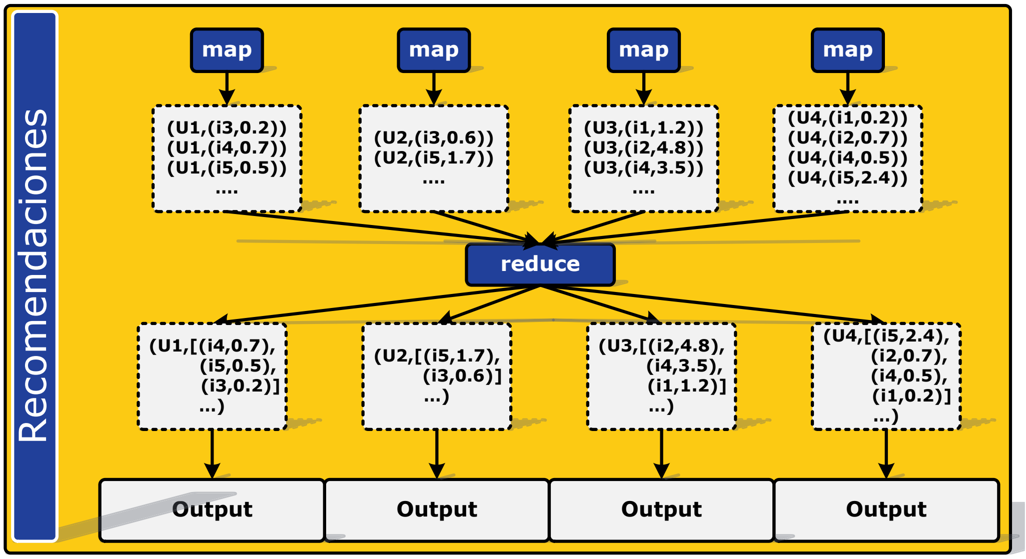


Figura 3.10. Esquema de recomendaciones de los usuarios

Para ver el esquema completo referirse al Apéndice A (Esquema map-reduce del sistema)

3.4. Diagrama de flujo de datos

Como hemos mencionado anteriormente, nuestro módulo está diseñado para operar manualmente, con una recurrencia mensual de horarios nocturnos, convirtiéndolo así en un procesamiento batch y offline. La justificación de este procedimiento se basa en el tiempo que tomaría procesar, en modo online, las recomendaciones de un usuario y sus similitudes con miles de otros usuarios que los sistemas pueden captar durante el día.

En la figura 3.11 se explica de manera gráfica el proceso de implementación del sistema de recomendaciones utilizando el paradigma map-reduce para las comparaciones, el framework de hadoop para el procesamiento síncrono y programado de tareas y las herramientas de Amazon EC2 que proveen el ambiente virtual necesario para un computo masivo y escalable de datos en la nube. En este diagrama se representa a las entidades (Operador, Usuario) en rectángulos de color azul, los procesos (P1, P2, ...etc) en círculos de color rojo y los almacenes (S3, HDFS) dentro de líneas continuas.

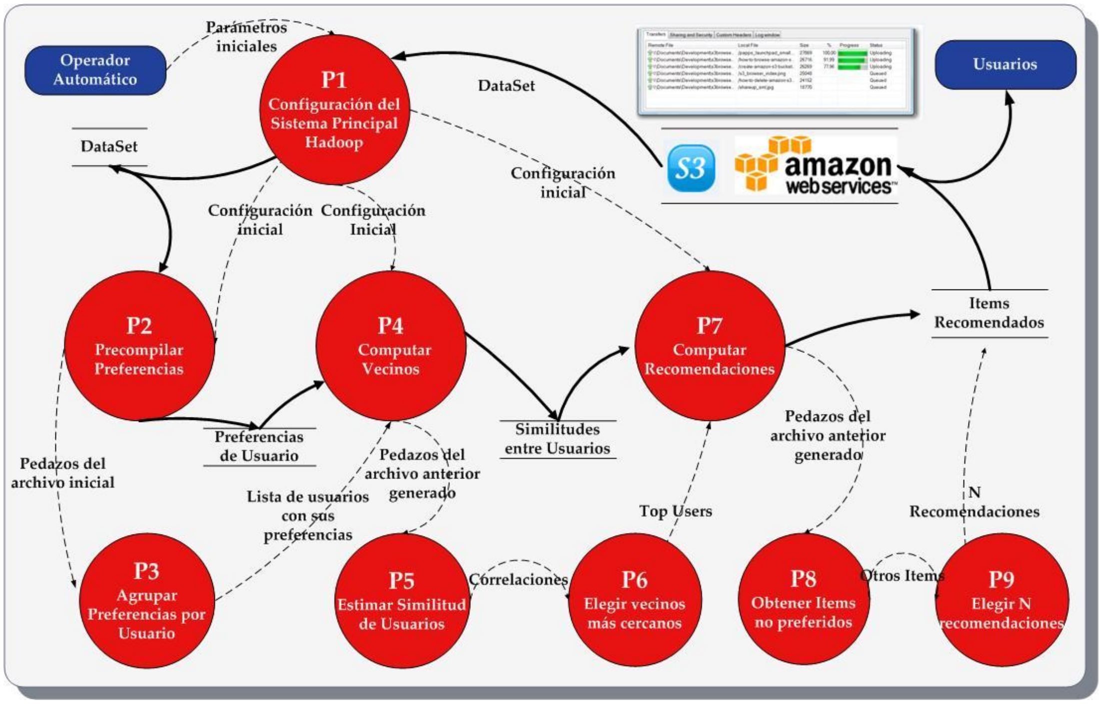


Figura 3.11. Diagrama de flujo de datos del sistema.

En resumen, este proceso tiene como objetivo la recomendación de N ítems nunca antes vistos para cada usuario utilizando la similitud de usuarios en base a la correlación de **Pearson** mencionada anteriormente.

El sistema inicia con el ingreso de los parámetros tales como el archivo de entrada, el archivo de salida, los N vecinos más cercanos y las N recomendaciones para cada usuario.

Luego pasa al proceso de pre-indexación de usuarios el cual genera un archivo de texto plano donde cada línea representa las preferencias por usuario. Este archivo será almacenado en el HDFS el cual sirve como entrada para el siguiente proceso que tiene como objetivo el cómputo de los N vecinos más cercanos por usuario. En la salida del cómputo mencionado se graban los N usuarios más similares con su valor de similitud por cada usuario activo.

Por último, el proceso de cómputo de las recomendaciones recibe como entrada el archivo de las similitudes para obtener los ítems nunca antes vistos por cada usuario y generar las N recomendaciones por medio de colas de prioridad, el cual dará como resultado un archivo en texto plano que luego será comprimido y subido al S3 para el almacenamiento respectivo (Ver figura 3.12).

En la sección 4.5, se explica con más detalle cada proceso del flujo de datos presentado anteriormente.

/musicalrecomendation/output/

File Name	File Size(KB)	Upload Time
simple10	0	
person10.tar.gz	101644	
simple10srmouti.tar.gz	108108	
simple10srmouti1-1.tar.gz	245474	
simple10srmouti1.tar.gz	237241	
simple10srmoutr.tar.gz	3282	
simple10srmouts.tar.gz	3889	
simple10srmoutu1.tar.gz	95995	
simple18.tar.gz	230958	

Figura 3.12. Captura de pantalla de los resultados almacenados en el S3.

CAPITULO 4

4. Implementación

4.1. Preparando el ambiente

Antes de iniciar la ejecución, se levanta el ambiente sobre el cual el programa se va a ejecutar. Para este propósito se contó con una máquina virtual de entrenamiento en Linux Ubuntu donde se tiene previamente instalado y/o configurado lo siguiente:

- Hadoop en su versión 0.18.3 al 27 de Enero del 2009.
- Certificados digitales y claves privadas de conexión con EC2. Para ver detalles de la instalación referirse al Apéndice B (Instalar los certificados digitales y claves privadas necesarias).
- Las herramientas de Amazon EC2. Para ver detalles de la configuración de EC2 referirse al Apéndice C (Instalar y configurar las herramientas de Amazon EC2).
- Conexión Ftp.
  - `# apt-get install vsftpd`
  - `# /etc/init.d/vsftpd start`

Luego de tener todas las herramientas mencionadas anteriormente, se cargan las librerías y los dataset respectivos al S3 mediante la herramienta del Firefox.

Teniendo cargado los archivos en el S3, se realiza el proceso de levantamiento del ambiente virtual en la nube de Amazon EC2 realizando lo siguiente:

1. Levantar el Cluster con la cantidad de nodos deseados en EC2:

   ```
   $HADOOP_HOME/bin/hadoop-ec2 launch-cluster <Nombre del cluster> <# nodos>
   ```

2. Conectarse al nodo master:

   ```
   ssh -i /home/training/.ec2/audioscrobbler.pem root@<id Nodo master>.compute-1.amazonaws.com
   ```

3. Bajar el ejecutable y el dataset del s3 al nodo master

   ```
   wget
   http://s3.amazonaws.com/musicalrecomendation/input/user_artist_data.txt
   wget http://s3.amazonaws.com/musicalrecomendation/srm.jar
   ```

4. Colocar el dataset bajado en el HDFS

   ```
   hadoop fs -put user_artist_data.txt
   ```

5. Ejecutar el programa con los parámetros de inicio

```
hadoop  jar  srm.jar
```

4.2. Definición de Parámetros de inicio

El sistema generador de las recomendaciones recibe los parámetros necesarios para llevar a cabo una operación que dará como resultado una salida que luego será utilizada en el siguiente proceso o "**Job**". Este esquema es posible mediante la utilización de la clase `JobConf` dada por el Framework de Hadoop.

Viendo el sistema de manera general lo que se quiere es enviar cuatro parámetros y al final obtener pedazos de archivos de recomendaciones por usuario generados por el último reduce.

Los parámetros de inicio son los únicos valores ingresados por el usuario operador antes de comenzar con las configuraciones de los jobs. Previamente se debe considerar subir el archivo inicial (dataset) al HDFS, debido a que este archivo debe estar disponible para todos los nodos participantes que realizarán el proceso de generación. La figura 4.1 muestra las entradas y salidas de todo el sistema en general.

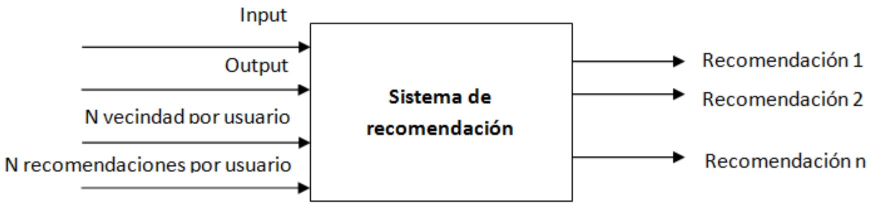


Figura 4.1. Diagrama de bloques de la vista general del sistema

4.3. Configuración de los jobs

Durante cada etapa de ejecución del programa, se configura un Job el cual define los siguientes elementos:

- Clases **map** y **reduce**. Ejemplo:
  - `jobConf.setMapperClass(UserSimilarityMapper.class)`
  - `jobConf.setReducerClass(UserNeighbourhoodReduccer.class)`
- Formato del archivo de entrada y salida. Ejemplo:
  - `jobConf.setInputFormat(TextInputFormat.class);`
  - `jobConf.setOutputFormat(TextOutputFormat.class);`
- Ruta del archivo de entrada y salida. Ejemplo:
  - `FileInputFormat.setInputPaths(jobConf, inputFilePath);`
  - `FileOutputFormat.setOutputPath(jobConf, outputPathPath);`
- Tipo de dato para clave/valor de salida del **map** y **reduce**. Ejemplo:

  - `jobConf.setMapOutputKeyClass(Text.class);`
  - `jobConf.setMapOutputValueClass(SimilarUserWritable.class);`
  - `jobConf.setOutputKeyClass(Text.class);`
  - `jobConf.setOutputValueClass(Text.class);`
- Variables de inicio clave/valor para el **map** o **reduce** (opcional). Ejemplo:
  - `jobConf.set(UserNeighbourhoodReduccer.NEAREST_NUSER_NEIGHBORHOOD, nvecindad);`
  - `jobConf.set(ItemsRecommendedReduccer.RECOMMENDATIONS_PER_USER, nrecomendaciones);`
- Carga de archivos en la cache distribuida (opcional). Ejemplo:
  - `DistributedCache.addCacheFile(new URI(inputPathDis), jobConf);`

4.4. Generación de recomendaciones utilizando valores de correlación de Pearson para las similitudes entre usuarios

Usando como referencia la figura 3.11, a continuación se describen los procesos para la generación de recomendaciones basados en usuarios:

Proceso 1: Configuración del Sistema Principal Hadoop

Al iniciar, el sistema solicita el ingreso de cuatro parámetros que servirán de configuración global de las tres tareas principales representados en los procesos 2, 4 y 7.

Estos parámetros corresponden a los siguientes campos <input> <output> <nvecindad> <nrecomendaciones>

donde:

<input> Es la ruta de origen HFDS donde se almacena el dataset.

<output> Es la ruta de destino HDFS donde se almacenan los resultados.

<nvecindad> Es el número de vecinos más cercanos.

<nrecomendaciones> Es el número de recomendaciones por usuario.

Cada tarea tiene como entrada la salida del proceso anterior cuyo almacén de datos corresponde al HDFS. Este es un repositorio temporal que se borra inmediatamente después de cerrar la instancia del nodo Maestro.

Proceso 2: Pre-compilar Preferencias

Se describe esta fase dentro del método llamado "`buildJobConf1`" el cual tiene como finalidad armar un archivo donde se almacenen todas las preferencias agrupadas por usuario en una línea de texto plano. Este resultado se guarda en el HDFS que servirá de entrada para el proceso de cómputo de los vecinos más cercanos.

Proceso 3: Agrupar Preferencias por Usuario

Mediante la función del **reducer**, se logra agrupar las preferencias por usuarios. Utilizando Writables personalizados se logra enviar objetos que luego serán agrupados por la clase UserReduccer. Así se genera un archivo que será almacenado en el HDFS para posterior uso.

Proceso 4: Computar Vecinos

En esta fase se configura el **map** y el reduce con las clases `UserSimilarityMapper` y `UserNeighbourhoodReduccer` respectivamente.

Dentro de este proceso también se configura la cache del HDFS para que almacene el archivo salida del proceso de pre-compilación descrito anteriormente, el cual contiene todas preferencias de los usuarios y sirve para comparar con cada valor de entrada de los **mappers**.

Proceso 5: Estimar Similitud de Usuarios

Para cumplir con este objetivo se utiliza la clase `UserSimilarityMapper`, el cual recibe como valor las preferencias de un usuario. Dicho usuario es comparado con cada uno de los usuarios almacenados en la cache con el fin de obtener un valor de cercanía basándose en la estimación de sus preferencias. Esto se lo obtiene a través de la referencia a la clase `Estimator` del paquete `NearestNUserNeighborhood` de la librería Mahout del proyecto Apache.

En la Figura 4.2 se muestra una porción de código que para cada usuario de entrada en el **map**, se va obteniendo el valor de similitud con otro usuario. La salida de este **map** tiene como clave el id del usuario y como valor un objeto que contiene el usuario cercano con su valor de similitud.

```
if(user2 != null && user1.compareTo(user2) !=
0){
     org.apache.mahout.cf.taste.neighborhood.Use
rNeighborhood .Estimator estimator = new

org.apache.mahout.cf.taste.neighborhood.UserNeig
hborhood .Estimator(new
PearsonCorrelationSimilarity(), user1,
Double.NEGATIVE_INFINITY);
     double similarity;

     try {
          mLogger.info("Creando similitud");
          similarity = estimator.estimate(user2);
          SimilarUserWritable suw = new
SimilarUserWritable(user1.getID().toString(),use
r2.getID().toString(),similarity);
          output.collect(new
Text(user1.getID().toString()), suw);
     } catch (Exception e) {
     e.printStackTrace();
     }

}
```

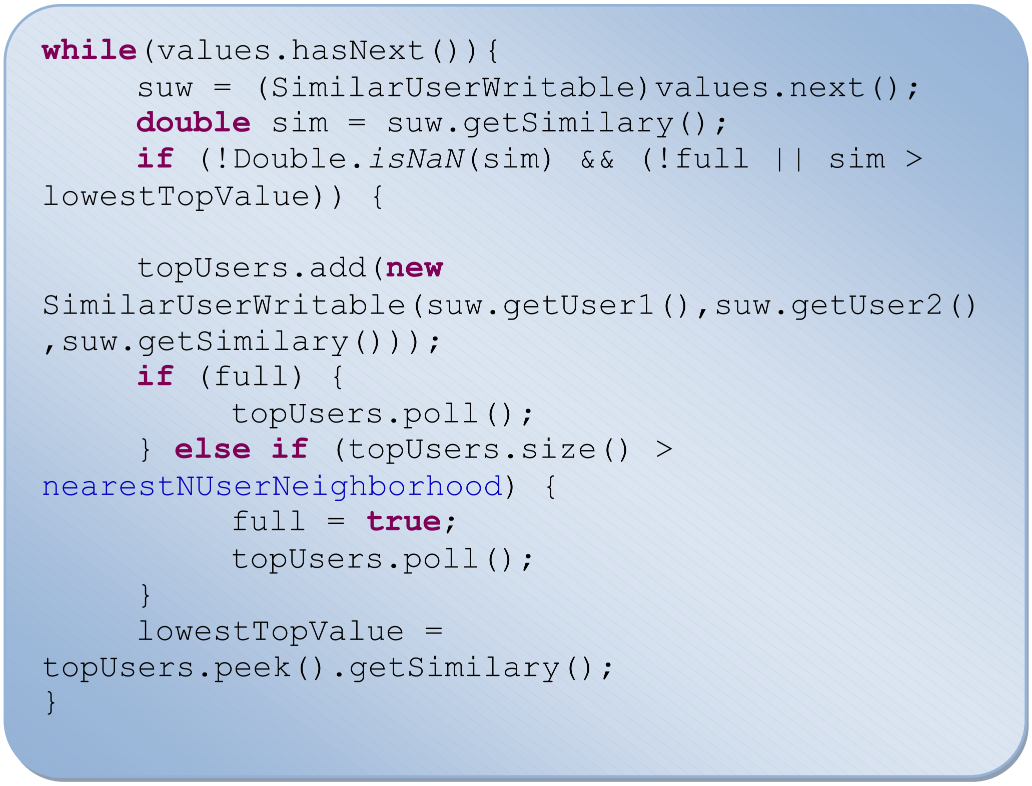
Figura 4.2. Código de estimación de similitudes

Proceso 6: Elegir vecinos más cercanos

Una vez obtenido todos los vecinos posibles de un usuario, se agrupan los N vecinos más cercanos de acuerdo a una cola de prioridad. Recordar que este número de vecinos fue especificado al inicio de la configuración del sistema. Ver proceso 1.

Esta tarea la realiza la clase `UserNeighbourhoodReduccer`, la cual agrupa todos los vecinos posibles de un usuario y los añade a una cola de prioridad siendo los de mayor puntaje los que permanezcan en esta lista.

En la Figura 4.3 se muestra el código de selección de los vecinos.



```
while(values.hasNext()){
     suw = (SimilarUserWritable)values.next();
     double sim = suw.getSimilary();
     if (!Double.isNaN(sim) && (!full || sim >
lowestTopValue)) {

     topUsers.add(new
SimilarUserWritable(suw.getUser1(),suw.getUser2()
,suw.getSimilary()));
     if (full) {
          topUsers.poll();
     } else if (topUsers.size() >
nearestNUserNeighborhood) {
          full = true;
          topUsers.poll();
     }
     lowestTopValue =
topUsers.peek().getSimilary();
}
```

Figura 4.3. Código de selección de los vecinos.

Proceso 7: Computar Recomendaciones

Este proceso corresponde al tercero y último Job del sistema de generación. Este computa las recomendaciones de todos los usuarios en base a los ítems que nunca se ha visto por parte de cada usuario.

Así como el proceso 4, se configura la cache del HDFS para que almacene el archivo salida del proceso de vecinos cercanos descrito anteriormente, el cual contiene todas los n vecinos de cada usuario. Así mismo se pasa a cada mapper el número de ítems que se desea recomendar para cada usuario.

Proceso 8: Obtener ítems no preferidos

Para realizar esta tarea, se necesitan dos archivos generados anteriormente por los dos jobs principales, es decir por el proceso 2 y 4. Con estos archivos se obtienen N vecinos más cercanos y preferencias para todo el umbral de usuarios.

Mediante la función `getAllOtherItems` de la librería Mahout y en conjunto con los archivos mencionados, se obtienen los ítems que no han sido preferidos por cada usuario y mediante la clase `Estimator` se generan valores de preferencia para cada ítem. Estos valores son enviados al **reducer** con clave iduser.

En la Figura 4.4 se presenta las iteraciones sobre cada uno de los ítems nunca antes visto por el usuario. Se generan valores de preferencia mediante la clase de Mahout `Estimator`.

```
while(i.hasNext()){
     Item itemid = (Item)i.next();

     try{
          double preferenceEstimated =
estimator.estimate(itemid);

          PreferenceWritable pw = new
PreferenceWritable(theUser.getID().toString(),ite
mid.getID().toString(),preferenceEstimated);

          output.collect(new
Text(theUser.getID().toString()), pw);

     } catch (Exception e) {
          e.printStackTrace();

     }
}
```

Figura 4.4. Código de generación de valores de preferencia para ítems nunca antes visto por un usuario.

Proceso 9: Elegir N recomendaciones

Este es el último proceso en donde se generan las recomendaciones para cada usuario. Aquí se agrupan por el key todas las preferencias de un usuario contenidos en la clase `PreferenceWritable`. De acuerdo con el parámetro inicial de N recomendaciones y por medio de colas de prioridad van quedando solo los de valor alto y se van eliminando los de valor bajo, logrando así la reutilización de variables y evitando el desbordamiento de la memoria en cada nodo levantado.

En la Figura 4.5 se muestra una parte del código donde se realiza la selección de los N ítems mejores postulantes a ser recomendados por cada usuario.

```
while(values.hasNext()){

     pref = (PreferenceWritable)values.next();

     double prefValue = pref.getPrefValue();

     if (!Double.isNaN(prefValue) && (!full ||
prefValue > lowestTopValue)) {

     topItems.add(new GenericRecommendedItem(new
GenericItem<String>(pref.getArtistID()),pref.get
PrefValue()));

    if (full) {
          topItems.poll();
    } else if (topItems.size() >
recommendationsPerUser){
          full = true;
          topItems.poll();
         }
      lowestTopValue =
topItems.peek().getValue();
     }
}
```

Figura 4.5. Código de selección de vecindad

4.5. Generación de recomendaciones utilizando valores de correlación simples (Producto Punto) para similitud entre usuarios

Con el objetivo de obtener tiempos más cortos en el proceso de recomendaciones, se ha implementado otro proceso que consiste en el cómputo de valores de similitud utilizando un algoritmo de correlación simple (Producto Punto) descritos en el Paper "*Pairwise Document Similarity in Large Collections with MapReduce*" [11].

Esta implementación conlleva a cambiar los mencionados procesos 4 y 5 del apartado anterior por los descritos a continuación:

Proceso 4: Agrupar usuarios por ítem

Mediante las clases `UserPrefFormaterMapper` y `UserPrefFromaterReduccer`, se logra agrupar usuarios por cada ítem musical. Utilizando la clase UserWritable, la función **map** envía como valor objetos `Writable` personalizados a la función reduce para que este los agrupen y formen una lista de usuarios por cada uno de los ítems.

Proceso 5: Agrupar por tuplas de usuarios

Este proceso recibe como entrada el archivo generado por la función anterior. Aquí, la clase `UserSimilarMapper` se encarga de formar claves a partir de tuplas de usuarios dentro de una clase llamada `SimilarUserWritable`, los cuales son asociados con el producto o multiplicación de valores de preferencia por el ítem de la referencia. Así el **mapper** genera tuplas correspondientes a pares de usuarios con un total de $\frac{m(m-1)}{2}$ pares, donde m es la longitud de la lista de usuarios participantes por cada ítem. Al final la clase `UserSimilarReduccer` agrupa todas las tuplas similares y suma toda la lista de valores por par para obtener el valor de similitud final.

En la figura 4.6 se muestra una parte del código donde se emiten las tuplas con su respectivo producto o multiplicación de sus preferencias sobre un ítem.

```
while(st.hasMoreTokens()){
     StringTokenizer stuser = new
StringTokenizer(st.nextToken());

     String user = stuser.nextToken();
     double pref =
Double.parseDouble(stuser.nextToken());
     StringTokenizer st2 = st;

     while(st2.hasMoreTokens()){
          StringTokenizer stuser2 = new
StringTokenizer(st.nextToken());
          String user2 = stuser2.nextToken();
          double pref2 =
Double.parseDouble(stuser2.nextToken());
          SimilarUserWritable suw = new
SimilarUserWritable(user,user2, pref*pref2);
          output.collect(suw, new
DoubleWritable(suw.getSimilary()));
          SimilarUserWritable suw2 = new
SimilarUserWritable(user2,user, pref*pref2);
          output.collect(suw2, new
DoubleWritable(suw.getSimilary()));
     }
}
```

Figura 4.6. Estimación de valores de correlación simple (Producto punto) para similitudes entre usuarios

Una vez obtenido el archivo de las tuplas con su valor de similitud, mediante el **mapper** *`GroupUserSimilarMapper`* se reagrupan los usuarios con todos sus vecinos posibles. La salida de esta función map sirve como entrada para el proceso 6 descrito en la sección 4.4 del apartado anterior sobre la elección de los N vecinos más cercanos.

A continuación en la figura 4.7 se muestra el diseño del esquema map-reduce para la obtención de los valores de similitud simple entre documentos basados en su contenido. Este diseño nos sirvió como referencia para la generación de los valores de correlación simple (Producto punto) entre usuarios y para la obtención de tiempos de respuesta más cortos. Aquí en el proceso de "Indexing", los **mappers** emiten como clave una letra o palabra dentro de un documento y como valor una tupla compuesta del id del documento acompañado del número de veces que la letra o palabra aparece dentro del documento. Al final del proceso de "Pairwise Similarity" se obtendrán archivos de texto plano en donde cada línea está compuesta por una tupla de dos documentos similares con su valor de correlación.

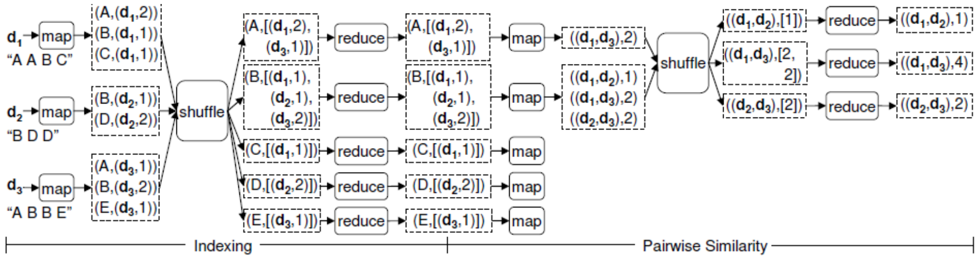


Figura 4.7. Computo de los valores de similitud simple presentado por Tamer Elsayed,_Jimmy Lin,† and Douglas W. Oard †[11]

Al finalizar todos los procesos descritos anteriormente, los archivos resultantes son manualmente subidos al S3 por medio de la herramienta de línea de comando s3cmd[16] (proyecto de código abierto bajo licencia pública GNU), el cual se usa para subir, recuperar y administrar la data en Amazon S3. Simplemente se lo instala y configura en el nodo maestro por medio de las siguientes líneas de comandos para sistemas operativos en Fedora:

```
$ yum install s3cmd

$ s3cmd --configure
```

---

[16] Michal Ludvig, 2008. Amazon S3 tools: s3cmd : command line S3 client. Available at: http://s3tools.org/s3cmd [Accedido Septiembre 4, 2009].

CAPITULO 5

5. Pruebas y Resultados

5.1. Pruebas

Para este propósito se adaptó el proceso de pruebas basados en el modelo en V (Verificación y Validación) dentro de un ciclo de desarrollo iterativo el cual asegura que desde las etapas más tempranas se vayan diseñando las pruebas. Además se utilizó como tipo las pruebas de Caja Blanca (pruebas unitarias) y Caja Negra (comprobación de valores límite, pruebas de situaciones de excepción, o pruebas de rendimiento del sistema).

5.2. Tiempos de respuesta

Nuestro primer inconveniente radicaba en la gran cantidad de memoria que utiliza la librería de Mahout para el almacenamiento de las características de todos los usuarios y en el tiempo de respuesta que llevaría la comparación de todos los ítems para todos los 150.000 usuarios del dataset en una máquina local.

Para los resultados del presente trabajo nos enfocamos en el tiempo de respuesta y costo de cada proceso ejecutado en EC2.

Se muestra la comparación del tiempo de ejecución del proceso de los vecinos más cercanos utilizando dos variantes en la obtención de los valores de similitud entre usuarios.

Para la ejecución de cada una de las dos variantes se levantaron 10 nodos con la misma configuración en su plataforma Linux e instalación de Hadoop versión 0.18.3-14. La figura 5.1 muestra los procesos o "Jobs" ejecutados para las dos variaciones.

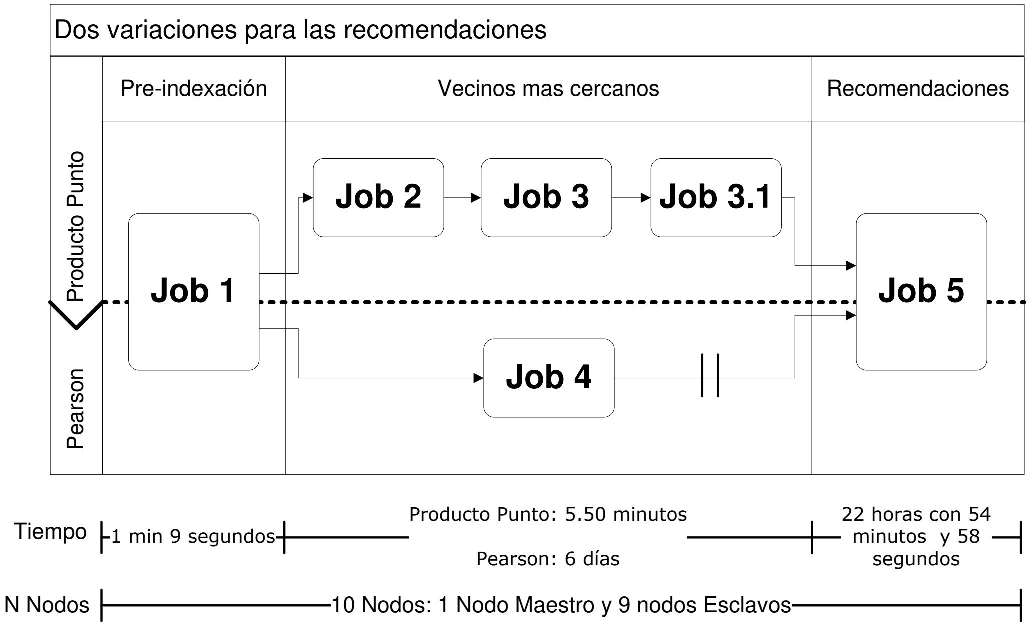


Figura 5.1. Diagrama de flujo de ejecución de los Jobs en 10 nodos.

Para el job 1, el cual es aplicado en las dos variaciones, se levantaron 4 tareas **map** y 10 tareas reduce, las cuales presentaron una duración total de ejecución de aproximadamente 1 minuto con 9 segundos.

Para el job 5, el cual es aplicado también para las dos variaciones, se levantaron 10 tareas **map** y 10 tareas **reduce**, los cuales presentaron una duración total de ejecución de aproximadamente 22 horas con 54 minutos y 58 segundos.

Para el tiempo de respuesta en la obtención de los vecinos más cercanos se tuvieron dos alternativas expuestas a continuación:

1. Aplicar correlación de **Pearson**

   Para este caso hacemos referencia al job 4, el cual tuvo un tiempo de respuesta mucho mayor, de aproximadamente 144 horas (6 días), es decir se computaba 1.500 usuarios de los 150.000 usuarios de dataset por 1.4 horas. Esta duración de tiempo fue estimada y obtenida de los tiempos de respuestas de los primeros porcentajes de avance en las tareas map. Por tanto el proceso fue terminado ya que la ejecución de todo el proceso habría implicado tener levantados los 10 nodos durante aproximadamente 6 días, lo cual habría causado un exceso en el presupuesto que era de $200.00 por grupo. Además esta decisión también es consecuencia de la poca cantidad de nodos compartidos disponibles para todos los proyectos del curso (20 nodos).

2. Aplicar una correlación Producto Punto

   Para resolver el problema anterior se tomó en consideración un procesamiento de los valores de similitud utilizando el agrupamiento de tuplas de usuarios similares y multiplicando las preferencias de sus ítems (Producto punto). Para ver más detalle de esta implementación referirse a la sección 4.5.

   Para este proceso se toman en cuenta los jobs 2, 3 y 3.1. A continuación se muestra una tabla de los resultados de la cantidad de tareas map-reduce y los tiempos incurridos por cada job:

Job	Tareas map	Tareas reduce	Tiempo map-reduce (min)
2	4	10	1.25
3	10	10	3.4
3.1	10	10	0.85
		Total	5.50

Tabla II. Tiempos de respuesta para la obtención de N vecinos más cercanos utilizando una correlación simple.

Aquí se puede apreciar que el tiempo de respuesta con respecto a la alternativa anterior se reduce en un 99.93%.

Este método no asegura que todos los usuarios tengan la vecindad requerida ni tampoco que contemple todos los ítems del data set porque en el proceso de agrupamiento de tuplas existen ítems que han sido escuchados por un sólo usuario. Como consecuencia, este usuario no debería participar como candidato a ser vecino. En el caso de la correlación de **Pearson**, sí se aseguraba de que todos los usuarios participen, ya que si había un usuario que poseía ítems no conocidos por todos los usuarios, el valor de similitud se infería y pasaba a formar parte de los posibles vecinos.

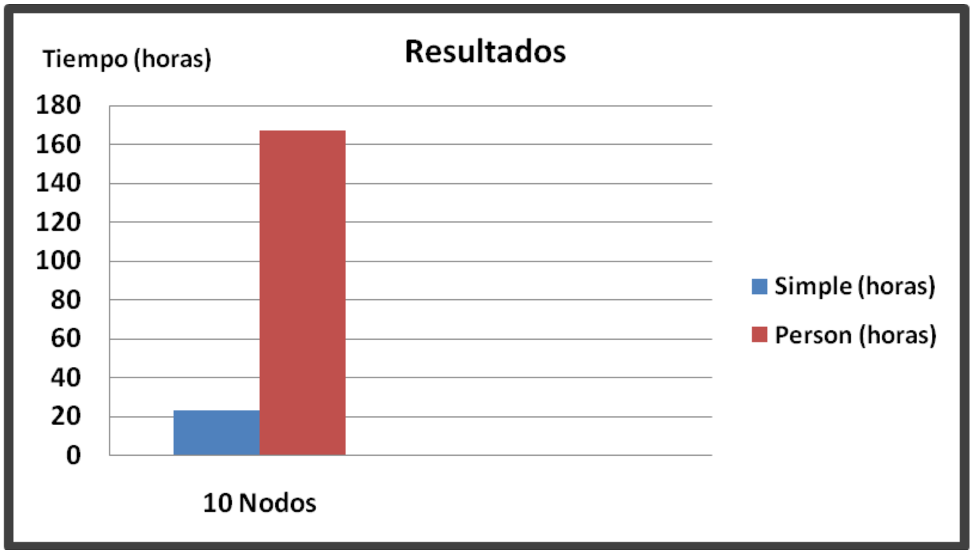


Figura 5.2. Gráfico de barras mostrando el tiempo de respuesta de los dos procesos de recomendaciones utilizando dos tipos de correlación.

5.3. Costos incurridos

Antes de describir los costos incurridos tanto en el almacenamiento de archivos en S3 como la ejecución de los jobs en EC2, se muestra una tabla de precios para el primero y un gráfico del cálculo de costo para segundo:

United States	
Storage	$0.150 per GB – first 50 TB / month of storage used
	$0.140 per GB – next 50 TB / month of storage used
	$0.130 per GB – next 400 TB /month of storage used
	$0.120 per GB – storage used / month over 500 TB
Data Transfer	$0.100 per GB – all data transfer in
	$0.170 per GB – first 10 TB / month data transfer out
	$0.130 per GB – next 40 TB / month data transfer out
	$0.110 per GB – next 100 TB / month data transfer out
	$0.100 per GB – data transfer out / month over 150 TB
Requests	$0.01 per 1,000 PUT, COPY, POST, or LIST requests
	$0.01 per 10,000 GET and all other requests*

Tabla III. Precios de petición, transferencia y almacenamiento en la cuenta de AWS S3[17] según la ubicación geográfica.

---

[17] Amazon Simple Storage Service (Amazon S3). Available at: http://aws.amazon.com/s3/ [Accedido Septiembre 2, 2009].

Estimación de costos usando la siguiente fórmula:

COSTO DE GRUPO = Σ COSTOS de TODAS LAS SESIONES.

COSTO DE UNA SESIÓN = $0.20 * # nodos en cluster * # de horas en que el cluster estuvo levantado (se redondean al entero superior) + $ 0.10 por GB transferido hacia el cluster + $ 0.17 por GB transferido desde el cluster

Ejemplo: El grupo XYZ levantó un cluster de 2 nodos y lo apagó dos y media horas después. Al día siguiente, levantó un cluster de 10 nodos y lo apagó 4.3 horas después.

COSTO SESIÓN 1 = $0.20 * 2 * 3 + $0.10 + $0.17 = $1.47

COSTO SESIÓN 2 = $0.20 * 10 * 5 + $0.10 + $0.17 = $10.27

COSTO TOTAL DEL GRUPO = $11.74

SALDO DISPONIBLE = $200 - $11.74 = $188.26

Figura 5.3. Documento del curso de cómo calcular costos de EC2.

Con esto, se estima que el proceso de recomendaciones utilizando la correlación de Pearson de la librería de Mahout tiene un costo mucho mayor que utilizando la correlación simple (Producto punto).

A continuación se muestra un cuadro donde se muestra el costo estimado utilizando correlaciones de Pearson y un costo real utilizando correlaciones simples levantando 10 nodos en EC2:

Descripción	Simple	Con Pearson
Constante (nodos)	$0.20	$0.20
Número de nodos en cluster	10	10
Duración cluster levantado (Horas)	24	169
Subtotal Nodos	$48.00	$338.00
Constante (Upload por GB)	$0.10	$0.10
Tamaño Upload (GB)	0.00000	0.00000
Subtotal Upload	$0.00	$0.00
Constante (Download por GB)	$0.17	$0.17
Tamaño Download (GB)	1	1
Subtotal Download	$0.17	$0.17
Costo Total Sesion	$48.17	$338.17

Tabla IV. Costos de ejecución de procesos en EC2

Al final de todo el proyecto incluyendo las pruebas realizadas y el almacenamiento, se ha estimado haber gastado unos $126.54 dólares quedando un saldo disponible de $73.46 dólares.

Categoría	Valor
Saldo Inicial	$200.00
Costo Total EC2 (US)	$125.73
Costo Total Storage (US)	$0.81
Costo Total Proyecto	$126.54
Saldo Disponible	$73.46

Tabla V. Costo Total del Proyecto

CONCLUSIONES Y RECOMENDACIONES

Conclusiones:

1. Los tiempos de ejecución de los algoritmos implementados aumentan con el cuadrado de la cantidad de usuarios y por ello presentan problemas de escalabilidad.
2. El costo del algoritmo de Pearson es tan alto en términos monetarios, que supera 8 a 1 al algoritmo basado en producto punto.
3. Aplicando producto punto no se obtienen recomendaciones para usuarios que han calificado ítems que otros usuarios no han calificado, Pearson resuelve este problema aplicando inferencias.

Recomendaciones:

1. La poca cantidad de nodos compartidos es una limitante. El producto punto se constituye en una alternativa viable para el procesamiento de las similitudes.
2. Para el seguimiento futuro de estos casos se puede buscar una alternativa al método de producto punto en el que se apliquen inferencias y utilice el esquema map-reduce para escalabilidad.

REFERENCIAS BIBLIOGRAFICAS

[1] Galán Nieto Sergio Manuel, *Filtrado Colaborativo y Sistemas de Recomendación*, España - Madrid: Universidad Carlos II de Madrid, 2007.

[2] Alton-Scheidl Roland, Ekhall Jesper, Van Geloven Olivier, Kovacs Laszlo & Micsik Andras, Lue Christopher, Messnarz Richard, Nichols David, Palme Jacob & Tholerus Torgny, Mason Dave & Procter Rob, Stupazzini Enrico & Vassali Massimo, y Wheeler Richard, "Social and Collaborative Filtering of Web Documents and News," Dagstuhl, Germany: Alfred Kobsa, GMD and Constantine Stephanidis, 1999, pág. 14.

[3] Saboya Vargas Aniceto Estudiante de doctorado en Inteligencia Artificial, "Uso de Recomendadores, Asistentes y Ayudantes en sistemas Tutores," *Universidad Politecnica de Cataluña*, pág. 6.

[4] Balabanović Marko, "Content-based, collaborative recommendation," *ACM New York, NY, USA*, vol. 40, Mar. 1997, pág. 72.

[5] Nichols David, "Implicit Rating and Filtering," *Proceedings of the*

5th DELOS Workshop on Filtering and Collaborative Filtering, Nov. 1997, pág. 5.

[6] Molina Valeria, Machado Rodrigo, y Betarte Leticia, "Recomendación de música basada en filtrado colaborativo," Grado, UdelaR - Carrera de Ingeniería en Computación.

[7] Velez-Langs Oswaldo y Santos Carlos, "Sistemas Recomendadores: Un enfoque desde los algoritmos genéticos," *Instituto de Investigación de Ingeniería Industrial de la UNMSM. (Peru)*, vol. 9, Jun. 2006, pág. 31.

[8] Dean Jeffrey and Ghemawat Sanjay. Google, Inc., "Google Research Publication: MapReduce," *Research Publications Google*, Dic. 2004, pág. 13.

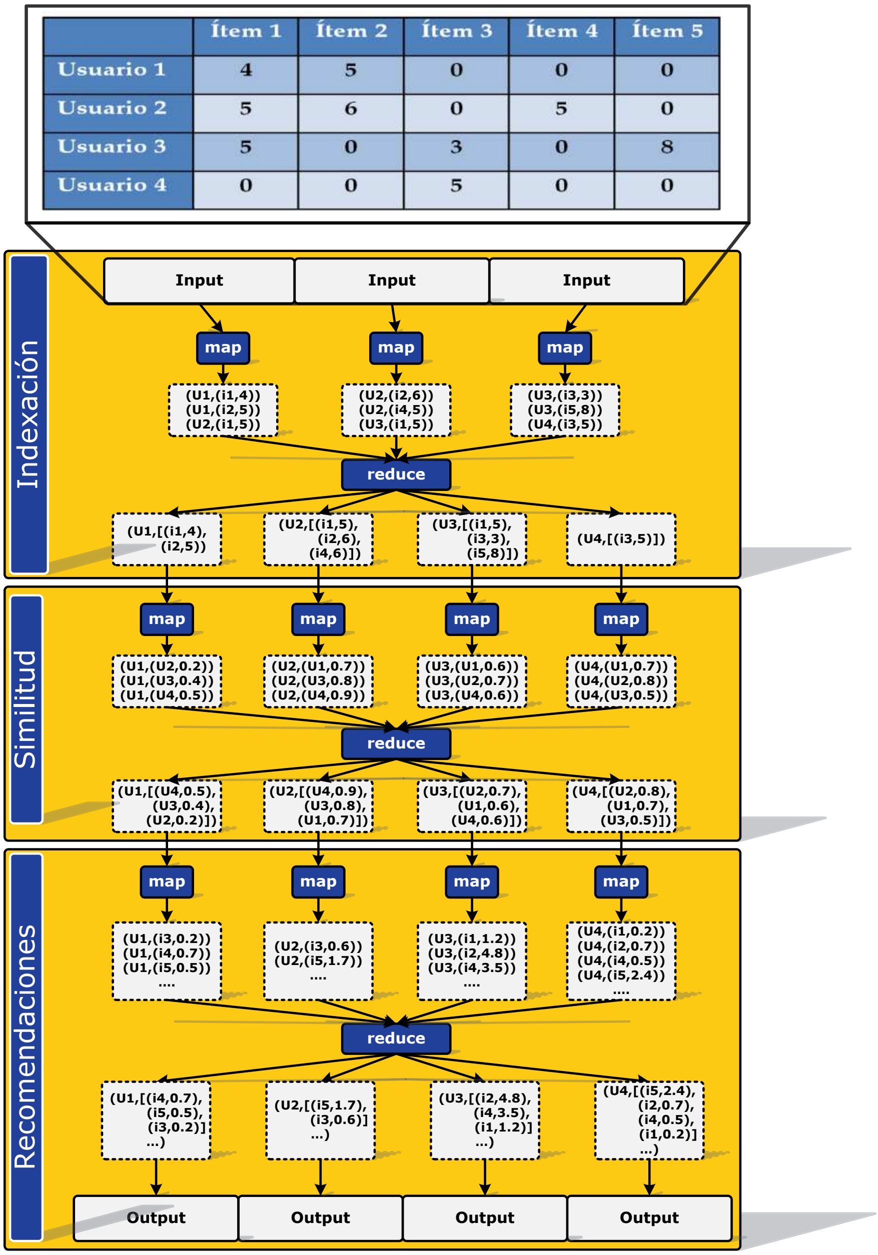
[9] "The Hadoop Distributed File System: Architecture and Design," *The Hadoop Distributed File System: Architecture and Design*, May. 2008.

[10] Vellino Andre, "PageRank Effect on Collaborative Filtering," *National Research Council (Ottawa, Ontario K1A 0R6)*, Jul. 2008, pág. 4.

[11] Elsayed, Oard Douglas, y Lin Tamer Jimmy, Pairwise Document Similarity in Large Collections with MapReduce, University of Maryland: Human Language Technology Center of Excellence and UMIACS Laboratory for Computational Linguistics and Information Processing, 2008.

APENDICE Y ANEXOS

Apéndice A: Esquema Map-Reduce general del Sistema



	Ítem 1	Ítem 2	Ítem 3	Ítem 4	Ítem 5
Usuario 1	4	5	0	0	0
Usuario 2	5	6	0	5	0
Usuario 3	5	0	3	0	8
Usuario 4	0	0	5	0	0

Apéndice B: Instalar los certificados digitales y claves privadas necesarias

En putty (conectado a la máquina virtual), o en una sesión de terminal de la máquina virtual:

```
cd $HOME

mkdir .ec2

date
```

Si la fecha y hora mostrada no es la actual, corregirlo de la siguiente manera:

```
sudo date -s "06/17/2009 16:17:45"
```

OJO: ¡en el comando anterior, deben usar la fecha/hora actual!

Con psftp (o utilizando alguna herramienta gráfica:

```
cd $HOME/.ec2/

put cert-X7WZ3CO4TTENSSCWQRMQ2YYNPBMIPZDF.pem

put pk-X7WZ3CO4TTENSSCWQRMQ2YYNPBMIPZDF.pem

put audioscrobbler.pem
```

En putty (conectado a la máquina virtual), o en una sesión de terminal de la máquina virtual:

```
cd .ec2

chmod 600 audioscrobbler.pem

cd ..
```

Apéndice C: Instalar y configurar las herramientas de Amazon EC2

En putty (conectado a la máquina virtual), o en una sesión de terminal:

```
wget http://s3.amazonaws.com/ec2-downloads/ec2-api-tools.zip
sudo apt-get install unzip
unzip ec2-api-tools.zip
wget http://cloudera-packages.s3.amazonaws.com/cloudera-for-hadoop-on-
ec2-0.4.0.tar.gz
gunzip cloudera-for-hadoop-on-ec2-0.4.0.tar.gz
tar –xvf cloudera-for-hadoop-on-ec2-0.4.0.tar
```

Lo siguiente se coloca en el .bashrc (por ejemplo, usando pico), para evitar que tengan que hacerlo para cada sesión:

```
export EC2_HOME=/home/training/ec2-api-tools-1.3-36506
export PATH=$PATH:$EC2_HOME/bin
export EC2_PRIVATE_KEY=/home/training/.ec2/pk-
X7WZ3CO4TTENSSCWQRMQ2YYNPBMIPZDF.pem
export EC2_CERT=/home/training/.ec2/cert-
X7WZ3CO4TTENSSCWQRMQ2YYNPBMIPZDF.pem
```

Configurar las variables de ambiente de EC2:

```
pico /home/training/cloudera-for-hadoop-on-ec2-0.4.0/bin/hadoop-ec2-env.sh
```

Comentario: “pico” es un editor de texto sencillo. En su lugar, pueden usar “vi” o su editor de preferencia. En ese archivo, deben hacer los siguientes cambios:

```
# Your Amazon Account Number.
AWS_ACCOUNT_ID=xxxx-xxxx-xxxx
# Your Amazon AWS access key.
AWS_ACCESS_KEY_ID=xxxxxxxxxxxxxxx
# Your Amazon AWS secret access key.
AWS_SECRET_ACCESS_KEY=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# Location of EC2 keys.
# The default setting is probably OK if you set up EC2 following
the Amazon Getting Started guide).
EC2_KEYDIR=`dirname "$EC2_PRIVATE_KEY"`
# The EC2 key name used to launch instances.
# The default is the value used in the Amazon Getting Started
guide.
KEY_NAME=audioscrobbler
# Where your EC2 private key is stored (created when following
the Amazon Getting Started guide).
# You need to change this if you don't store this with your
other EC2 keys.
PRIVATE_KEY_PATH=`echo "$EC2_KEYDIR"/"$KEY_NAME.pem"`
```

Apéndice D: Lista de los ítems musicales (Artistas) que fueron más recomendados durante todo el proceso.

N veces recomendado	Id Artista	Nombre Artista/Grupo
26479	1000597	Bon Jovi
12449	1000591	The Chemical Brothers
12070	5793	The Distillers
11389	722	Lagwagon
10533	729	Missy Elliott
9965	721	Goldfinger
9930	728	Ministry
8199	1002622	Bonnie Tyler
7203	1002621	Erasure
7174	1002623	Haddaway
7092	1002624	Berlin
6893	4371	Lostprophets
6609	5792	The Damned
6380	1007903	Maroon 5
6280	1006029	Converge
5870	5798	The Doobie Brothers
5756	1002626	N-Trance
5332	1000857	SR-71
4640	4377	Ludwig van Beethoven
4585	1244415	Murphy Lee