Algoritmo paralelo para detectar sucesos en noticias en línea *

Reynaldo Gil-García†, José Manuel Badía-Contelles‡ y Aurora Pons-Porrata†

Resumen— **En este trabajo se expone un algoritmo paralelo que detecta los sucesos en un flujo de noticias en línea. Este algoritmo usa el paradigma de programación paralela de paso de mensajes sobre el estandar MPI, con el objetivo de paralelizar el algoritmo de agrupamiento compacto incremental. Como resultado se obtiene un algoritmo eficiente y portable, que logra un distribución adecuada de los datos y de los cálculos entre los procesadores. En el trabajo se analiza experimentalmente el comportamiento del algoritmo en un cluster de computadoras personales, aplicandose a una colección de noticias del periódico El País. Se muestra que con el algoritmo paralelo obtenido es posible detectar sucesos en un gran volumen de noticias en línea.**

Palabras clave— **detección de sucesos en línea, algoritmo de agrupamiento, algoritmo paralelo.**

I. Introducción

En aplicaciones donde existe un flujo ininterrumpido de documentos se requieren mecanismos automáticos que operando a la misma velocidad que el flujo, organicen y filtren la información para su posterior estudio por parte de los usuarios. Una de estas aplicaciones consiste en la detección y el seguimiento automático de sucesos en flujos de noticias digitales, también conocida como TDT (Topic Detection and Tracking)[1]. El principal problema planteado consiste en determinar si un documento entrante informa sobre un nuevo suceso o forma parte de otros sucesos recogidos por el sistema. La tarea de detección es una abstracción experimental del agrupamiento de noticias. El objetivo de un sistema de detección es agrupar las noticias que abordan un mismo suceso.

En esta aplicación hay que tener muy en cuenta que el conjunto de noticias cambia en el tiempo, pues es necesario modificar el agrupamiento a medida que se van publicando nuevas noticias para mantenerlo actualizado. Por tanto, se requieren algoritmos que sean capaces de realizar la actualización a medida que se publican las nuevas noticias, sin necesidad de empezar nuevamente desde el principio. Estos algoritmos reciben el nombre de algoritmos incrementales o en línea.

Con el creciente tamaño de la cantidad de noticias disponibles en línea, una sola computadora no puede resolver el problema de la búsqueda de los sucesos de una parte significativa de las fuentes de información disponibles en un tiempo aceptable. Además, muchas veces una computadora tampoco tiene la memoria necesaria para almacenar los datos que necesita el algoritmo, por lo que hay que recurrir a datos en disco, lo que hace más lento aún el procesamiento. Esta es una situación característica donde se puede intentar resolver el problema explotando el poder de cómputo y la memoria disponible en un conjunto de procesadores. Así, la paralelización de los algoritmos de detección es la opción para obtener en un tiempo razonable los sucesos de una parte significativa de las noticias existentes en línea.

En el trabajo se expone la paralelización de un algoritmo de agrupamiento incremental que busca los conjuntos compactos existentes en un flujo de noticias en línea. El algoritmo paralelo obtenido logra un adecuado balance de carga entre los procesadores al repartir equitativamente los objetos entre los procesadores y lograr que cada procesador realice aproximadamente el mismo trabajo. Los resultados experimentales obtenidos demuestran la validez de la paralelización realizada.

El trabajo se estructura como sigue. En la sección 2 se explican las características del problema a resolver y se expone la versión secuencial del algoritmo de agrupamiento compacto incremental. La sección 3 detalla el algoritmo paralelo propuesto. En la sección 4 presentamos una valoración de los resultados experimentales obtenidos al aplicar el algoritmo a la detección de sucesos en un flujo de noticias de un periódico digital. Finalmente, damos las conclusiones de nuestro trabajo.

II. Especificación del problema

Los elementos esenciales a tener en cuenta para resolver el problema de la detección de sucesos en flujos de noticias en línea son: la forma de representación de las noticias, la función de semejanza entre noticias y el algoritmo de agrupamiento a utilizar para formar los sucesos.

A. Representación de las noticias

Las noticias de entrada en nuestro sistema forman parte de periódicos digitales en formato XML. Éstas han sido generados a partir de las versiones HTML disponibles en línea a través de la Web siguiendo el proceso descrito en [2]. Los documentos XML obtenidos conservan la estructura lógica de la noticias, de tal forma que es posible distinguir sus distintas componentes, como por ejemplo, el titular, los autores, el lugar de redacción, etc. Para este trabajo sólo consideraremos el contenido textual de las noticias y su fecha de publicación.

Una noticia d^i se representa por dos componentes:

- Un vector de términos con sus frecuencias relativas de aparición en el texto: $T^i = (TF_1^i, TF_2^i, ..., TF_n^i)$, donde TF_j^ies la frecuencia

*Supported by the Spanish CICYT project TIC2002-04400-C03-01.

† {gil,aurora}@app.uo.edu.cu, Universidad de Oriente, Cuba.

‡ badia@icc.uji.es, Universidad Jaume I, Castellón, España.

relativa del término $t_j$en la noticia d^i. Cada término representa la forma reducida de un conjunto de palabras, por ejemplo, todas las conjugaciones de un verbo se representará con su infinitivo, el plural de una palabra por su forma singular, etc. No se considerarán como términos aquellas palabras con escaso contenido semántico, como son las preposiciones, adverbios, conjunciones, artículos, etc.
- La fecha de publicación de la noticia expresada en el calendario gregoriano: FP^i.

B. Medida de semejanza entre noticias

Todo proceso automático de agrupamiento de noticias, incluido la detección de sucesos, está basado en una medida de semejanza. El objetivo de la medida de semejanza es distinguir si dos noticias abordan o no un mismo suceso. En la mayor parte de los algoritmos presentados en la literatura, la medida de semejanza se basa en el coseno de los vectores que representan los documentos. En nuestro trabajo, utilizamos una medida de semejanza que tiene en cuenta no solamente la medida del coseno sino también la proximidad temporal de las noticias. Para ello, definimos la función *dist* como el tiempo transcurrido entre la publicación de las noticias que se comparan y el umbral δ que representa el tiempo máximo requerido para considerar si dos noticias abordan o no el mismo suceso. De esta forma, consideramos que dos documentos pertenecen al mismo suceso si su contenido y sus fechas de publicación son cercanas.

Teniendo en cuenta lo anterior, la definición de la función de semejanza entre noticias es como sigue:

$$S(d^i, d^j) = \left\{ \begin{array}{cc} \cos(d^i, d^j) & si\, dist(FP^i, FP^j) \leq \delta \\ 0 & en\, otro\, caso \end{array} \right\}$$

C. Algoritmo Compacto Incremental

La mayoría de las técnicas desarrolladas hasta la fecha para la detección de sucesos están basadas en los clasificadores automáticos provenientes del área de la Recuperación de Información. Así, en [3] se propone un algoritmo similar al *Single-Pass* [4] para la detección de sucesos. En [5] se usa una variante incremental del algoritmo *K-means*, donde no se necesita de antemano fijar el número de grupos k, y en [6] se aplica el algoritmo *Fractionation* [7] utilizando *Group-average* (GAC) sobre los documentos que llegan a una determinada ventana de tiempo. Estos trabajos requieren una definición del centro de un grupo, por lo que producen grupos aproximadamente esféricos. Además los grupos obtenidos dependen del orden de presentación de las noticias, lo que consideramos una característica indeseable, pues los grupos existentes no dependen del orden en que se presenten las noticias, sino que sólo dependen de las características internas de los datos.

El algoritmo de agrupamiento compacto incremental [8] crea particiones del conjunto de datos. Su principal ventaja sobre los algoritmos incrementales existentes radica en que el agrupamiento obtenido es independiente del orden de presentación de los documentos. Además, no realiza una asignación irrevocable de los documentos a los grupos, lo que permite que errores cometidos al principio cuando se dispone de poca información puedan ser corregidos. Otra ventaja del mismo es que no restringe la forma de los grupos. Por otro lado, presenta un efecto de encadenamiento restringido que intuitivamente coincide con la forma manual de agrupar a las noticias en sucesos. Este algoritmo se ha utilizado con éxito para la detección de sucesos en línea en colecciones de noticias digitales [9].

El algoritmo compacto incremental es un algoritmo basado en grafos. Dos documentos (noticias) cuya semejanza es mayor o igual que un cierto umbral β_0 (definido por el usuario) se denominan β_0-semejantes. Se llama grafo de máxima β_0-semejanza ($max - S$) al grafo orientado donde los vértices son los documentos a agrupar y existe un arco del vértice d^i al vértice d^j si se cumple que d^j es el documento más β_0-semejante a d^i. Este algoritmo de agrupamiento obtiene de forma incremental los conjuntos compactos de una colección de documentos. Los conjuntos compactos coinciden con las componentes conexas del grafo $max - S$ sin tener en cuenta la orientación. En la figura 1 mostramos un grafo $max - S$, donde, por ejemplo, la noticia d^1es la más β_0-semejante a d^0. En el grafo de la izquierda, los conjuntos compactos son: $\{d^0, d^1, d^2, d^3\}$ y $\{d^4\}$.

El algoritmo compacto almacena para cada documento el valor de su máxima β_0-semejanza y el conjunto de los documentos conectados con él en el grafo $max - S$. Cada vez que se presenta un nuevo documento (d), se calcula su semejanza con los documentos de los grupos existentes y se actualiza el grafo. La llegada del nuevo documento puede provocar cambios en el agrupamiento existente, pues algunos de los conjuntos compactos existentes pierden esta propiedad y surgen otros nuevos. Por tanto, al finalizar la actualización del grafo de máxima β_0-semejanza se reconstruyen los conjuntos compactos a partir del nuevo documento y de los documentos que pertenecen a los grupos que pueden perder la propiedad de ser compacto. Los conjuntos compactos que no tienen documentos conectados con el nuevo permanecen inalterables.

Durante la actualización del grafo, el algoritmo construye los siguientes conjuntos:

- *GruposAProcesar*: Un grupo pertenece a este conjunto si tiene algún documento d' que cumple las condiciones siguientes: 1) d es el más β_0-semejante a d' y los documentos existentes que eran sus más β_0-semejantes dejan de serlo. 2) d' tenía al menos dos documentos que eran sus más β_0-semejantes o d' es el más β_0-semejante a al menos un documento de ese grupo. A este conjunto pertenecen los grupos que potencialmente pueden dejar de ser compactos cuando se eliminan de ellos a los documentos d' que cumplen las condiciones anteriores y, por lo tanto, deben ser reconstruidos.

- *DocumentosAUnir*: Un documento d' pertenece a este conjunto si cumple las condiciones siguientes: 1) d es el más β_0-semejante a d' y el único documento más β_0-semejante a d' deja de serlo. 2) d' no es el más β_0-semejante a ningún documento de su grupo. Los documentos de este conjunto pertenecerán al mismo conjunto compacto que d.
- *GruposAUnir*: Un grupo pertenece a este conjunto si no pertenece a *GruposAProcesar* y tiene al menos un documento d' que cumple una de las condiciones siguientes: 1) d' es el más β_0-semejante a d. 2) d se incorpora al conjunto de documentos más β_0-semejantes a d', es decir, d está conectado con d' y no se rompe ningún arco de d' en el grafo $max-S$. Todos los documentos de estos grupos pertenecerán al mismo conjunto compacto que el nuevo documento.

Los pasos del algoritmo se muestran a continuación.

1. Llegada del nuevo documento d.
2. Actualización del grafo $max-S$.
   Para cada documento de los grupos existentes se calcula la semejanza con el nuevo documento, se actualiza el valor de su máxima β_0-semejanza y el conjunto de los documentos conectados con él en el grafo $max-S$. Se calcula la máxima β_0-semejanza del nuevo documento y el conjunto de los documentos conectados con él. Se crean los conjuntos *GruposAProcesar*, *DocumentosAUnir* y *GruposAUnir*. Cada vez que se incorpora un documento a *DocumentosAUnir* se elimina del grupo al que pertenecía.
3. Reconstrucción de los conjuntos compactos.
   Sea C el conjunto formado por el nuevo documento y todos los documentos que pertenecen a los grupos de *GruposAProcesar*. Formar los conjuntos compactos existentes entre los documentos de C y añadirlos a la lista de grupos existentes. Añadir al conjunto compacto donde pertenece el nuevo documento, todos los documentos del conjunto *DocumentosAUnir* y todos los documentos que pertenecen a los grupos que están en el conjunto *GruposAUnir*. Eliminar los grupos de *GruposAProcesar* y de *GruposAUnir* de la lista de grupos existentes.

Este algoritmo tiene una complejidad temporal de $O(n^2)$. Sin embargo, si se utiliza la medida de semejanza definida en II-B, la función *dist* y el umbral δ definen una ventana temporal, por lo que cuando se presenta el nuevo documento sólo es necesario compararlo con los k documentos que caen dentro de esta ventana. Por tanto, sólo pueden modificarse los conjuntos compactos que incluyen a dichos documentos. La complejidad temporal del algoritmo usando esta función de semejanza se reduce, entonces, a $O(kn)$.

III. Algoritmo Compacto Incremental Paralelo

Se han desarrollado paralelizaciones de otros algoritmos de agrupamiento como el *K-Means* [10] y el GLC [11]. El *K-Means* paralelo no es directamente aplicable a la detección de sucesos debido a la naturaleza incremental de este problema. El algoritmo GLC busca de forma incremental las componentes conexas en grafos de semejanzas, por lo que su paralelización podría ser utilizada directamente para detectar los sucesos. Sin embargo, este algoritmo presenta un elevado efecto de encadenamiento, lo que provoca que noticias muy poco relacionadas entre sí se ubiquen en el mismo suceso, por lo que no se considera satisfactorio para esta aplicación.

Para paralelizar el compacto incremental suponemos una arquitectura de computadora paralela con memoria distribuida por lo que utilizamos el paradigma de paso de mensajes [12]. Dentro de este paradigma utilizamos la estrategia de particionado de datos y el esquema de maestro-esclavos. En distintas fases del algoritmo un procesador actuará como maestro y los restantes, como esclavos.

En la paralelización del algoritmo compacto repartimos uniformemente los datos entre los procesadores de forma tal que el procesador i almacena la descripción del documento j si el resto de la división de j entre p (cantidad de procesadores) es i. Con esto, tratamos de lograr un equilibrio de carga entre los procesadores, lo cual es una característica deseable en todo algoritmo paralelo. Además de la descripción de sus documentos, cada procesador mantiene para cada documento el valor de su máxima β_0-semejanza y los índices de los documentos conectados con él en el grafo $max-S$. Cada procesador mantiene también una lista de grupos (conjuntos compactos) y, a su vez, de cada grupo almacena una lista de los documentos del procesador que pertenecen a ese grupo. No necesariamente todos los procesadores contienen a todos los grupos, pues un procesador sólo tiene los grupos a los que pertenecen sus documentos según la distribución de carga realizada.

Ante la llegada de un nuevo documento d, cada procesador calcula la semejanza de sus documentos con d y actualiza la parte del grafo que involucra a sus documentos, lo cual requiere un intercambio de información entre los procesadores. En este proceso se determina qué grupos pueden perder la propiedad de ser compactos. Durante la actualización del grafo, cada procesador construye los conjuntos *GruposAProcesar*, *DocumentosAUnir* y *GruposAUnir* de forma similar al caso secuencial. Además, cada procesador crea el conjunto *ArcosPerdidos* que contiene los pares de documentos (d', d''), donde d' es un documento de este procesador y d'' es un documento de otro procesador de forma tal que el arco existente entre ellos se rompió. Este rompimiento se produce debido a que d es el más β_0-semejante a d' y d'' deja de serlo.

Al finalizar la actualización del grafo de máxima β_0-semejanza, se reconstruyen los conjuntos compactos a partir de d y de los documentos que pertenecen a los grupos que pueden perder la propiedad de ser compactos. Los conjuntos compactos que no tienen documentos conectados con d permanecen inaltera-

dos. Para ello, cada procesador, a partir de sus documentos que pertenecen a los grupos que pueden perder la propiedad de ser compactos, construye los subconjuntos de compactos que pueden formarse. Para cada uno de estos subconjuntos se forma el conjunto *Llegada,* que contiene a todos los documentos que están conectados con algún documento de ese subconjunto y no pertenecen a dicho subconjunto. Nótese que al conjunto *Llegada* siempre pertenecerán documentos de otro procesador. Luego, el procesador 0 conforma los nuevos conjuntos compactos a partir de estos subconjuntos y sus conjuntos *Llegada* asociados. Si un subconjunto compacto tiene al menos un documento que pertenece al conjunto *Llegada* asociado a otro subconjunto compacto, entonces estos dos subconjuntos forman un mismo conjunto compacto. Aplicando esta propiedad se construyen los conjuntos compactos. Nótese que cuando se unen dos subconjuntos compactos, es necesario formar también su conjunto *Llegada* asociado.

Los pasos del algoritmo se muestran a continuación:

1. El procesador p_0 difunde la descripción del nuevo documento d.
2. Actualización del grafo $max - S$.
   - En cada procesador p_i:
     Para cada uno de sus documentos se calcula la semejanza con d, se actualiza el valor de su máxima β_0-semejanza y el conjunto de los documentos conectados con él en el grafo $max - S$. Se crean los conjuntos *GruposAProcesar*, *DocumentosAUnir* y *ArcosPerdidos*. Cada vez que se incorpora un documento a *DocumentosAUnir* se elimina del grupo al que pertenecía. Se crea el conjunto *GruposAUnir* formado inicialmente por los grupos donde existen documentos de este procesador para los cuales d se incorpora a su conjunto de documentos más β_0-semejantes. Se crean los conjuntos AEl_i = conjunto de los documentos de este procesador de los cuales d es el más β_0-semejante y $DeEl_i$ = conjunto de sus documentos que son más β_0-semejantes a d. Se calcula además la máxima β_0-semejanza de d con los documentos del procesador ($MaxSem_i$).
   - Unión entre todos los procesadores de sus conjuntos *GruposAProcesar* para formar *GruposAProcesarGlobal*.
   - Distribución entre todos los procesadores de *ArcosPerdidos* de forma tal que $ArcosPerdidosFinal_j = (d', d'')/(d', d'') \in \cup ArcosPerdidos_i \wedge d'' \in p_j$, donde p_j es el j-ésimo procesador, $i, j = 0, ..., p-1$.
   - Recogida en el procesador dueño de d de AEl_i,$DeEl_i$ y $MaxSem_i$,$i = 0, ..., p-1$.
   - En el procesador dueño de d:
     $MaxSemd$ = máx$\{MaxSem_i\}$, $i = 0, ..., p-1$. Almacenar los documentos conectados con d, los cuales serán todos los documentos pertenecientes a $\cup AEl_i$, $i = 0, ..., p-1$ o a $\cup DeEl_j$, tal que $MaxSem_j = MaxSemd$ y $0 \leqslant j \leqslant p-1$.
   - Difusión desde el dueño de d de $MaxSemd$
   - En cada procesador p_i:
     Si $MaxSem_i = MaxSemd$
     - Conectar a todos los documentos de $DeEl$ con d.
     - Agregar a *GruposAUnir* los grupos de los documentos que pertenecen a $DeEl$.
     - $GruposAUnir = GruposAUnir \setminus GruposAProcesarGlobal$.

     Eliminar del subgrafo $max - S$ los arcos que estén en $ArcosPerdidos_i$.
3. Reconstrucción de los conjuntos compactos.
   - En cada procesador:
     C = conjunto de sus documentos pertenecientes a los grupos de *GruposAProcesarGlobal*. Si es el procesador dueño de d, agregar d al conjunto C. Formar los subconjuntos de compactos existentes en C. Para cada subconjunto compacto formado crear su conjunto *Llegada* asociado.
   - Unión entre todos los procesadores de sus conjuntos *GruposAUnir* para formar *GruposAUnirGlobal*.
   - Recogida en el procesador 0 de los subconjuntos de compactos formados y sus conjuntos *Llegada* asociados.
   - En el procesador 0:
     Se construyen los conjuntos compactos a partir de los subconjuntos de compactos formados en cada procesador y sus conjuntos *Llegada* asociados.
   - Difundir desde el procesador 0 los conjuntos compactos obtenidos.
   - En cada procesador:
     Actualizar sus grupos. Agregar al grupo de d los objetos que pertenecen a los grupos de *GruposAUnirGlobal* o a su conjunto *DocumentosAUnir*. Eliminar los grupos que pertenecen a *GruposAUnirGlobal* o a *GruposAProcesarGlobal*.

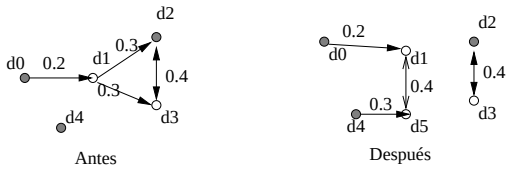

Fig. 1. Grafo $max - S$ antes y después de procesar a d^5.

Para ayudar a la comprensión del algoritmo, en la figura 1 mostramos el grafo $max - S$ antes de presentarse el documento d^5 y después de procesarlo. En el mismo cada arco está etiquetado con el valor de la semejanza entre los documentos y los vértices de documentos que pertenecen al procesador 0 están rellenos en negro. Si suponemos que tenemos dos procesadores, los datos en cada uno de ellos antes de iniciar el procesamiento de d^5 se muestran en la ta-

TABLA I
DATOS ANTES DE PROCESAR A d^5

Procesador	0	1
Documentos	d^0,d^2,d^4	d^1,d^3
Grupos	$G_0 = \{0, 2\}$ $G_1 = \{4\}$	$G_0 = \{1, 3\}$

TABLA II
DATOS PARA ACTUALIZAR EL GRAFO.

Procesador	0	1
GruposAProcesar	Ø	$\{G_0\}$
DocumentosAUnir	Ø	Ø
ArcosPerdidos	Ø	$\{(d^1, d^2)\}$
AEl	$\{d^4\}$	$\{d^1\}$
DeEl	$\{d^4\}$	$\{d^1\}$
MaxSem	0,3	0,4

bla I. Cuando se agrega d^5, se procede a actualizar el grafo. Para ello, se construyen los conjuntos *GruposAProcesar*, *DocumentosAUnir*, *ArcosPerdidos, AEl* y *DeEl*, los cuales se muestran en la tabla II. En el procesador 1 (dueño del documento d^5) se recogen *GruposAProcesar*, *ArcosPerdidos, AEl* y *DeEl* para obtener $GruposAProcesarGlobal = \{G_0\}$ y difundirlo, crear los enlaces de d^5, enviar al procesador 0 el arco (d^1, d^2) y difundir $MaxSemd = 0{,}4$. Con esta información cada procesador actualiza su parte del grafo, busca los subcompactos existentes entre sus objetos que pertenecen a los grupos de *GruposAProcesarGlobal* y d^5 y crea los conjuntos de llegada de cada subcompacto. Lo resultados que se obtienen en cada procesador se muestran en la tabla III. Estos resultados se recogen en el procesador 0 que los procesa y determina que los subcompactos S_0 de los procesadores 0 y 1 forman un solo compacto, pues el conjunto *Llegada* L_0 del procesador 0 contiene documentos del conjunto S_0 del procesador 1. Identicamente determina que los subcompactos S_1 de ambos procesadores son un mismo conjunto compacto. A cada conjunto compacto obtenido se le asigna un identificador y se difuden estos resultados. Cada procesador actualiza sus compactos, en particular el procesador 0 añade d^4 (que pertenece a G_1 que está en *GruposAUnir*) al grupo del d^5 y se elimina G_1.

Cabe destacar algunas características del algoritmo paralelo diseñado, como son:

TABLA III
DATOS PARA CREAR LOS COMPACTOS

Procesador	0	1
Subcompactos	$S_0 = \{0\}$ $S_1 = \{2\}$	$S_0 = \{1, 5\}$ $S_1 = \{3\}$
Llegada	$L_0 = \{1\}$ $L_1 = \{3\}$	$L_0 = \{0, 4\}$ $L_1 = \{2\}$
GruposAUnir	$\{G_1\}$	Ø

1. Existe un equilibrio de carga entre los procesadores, pues cada procesador tiene aproximadamente $\frac{n}{p}$ documentos y cada uno calcula aproximadamente la misma cantidad de semejanzas. Además el grafo de máxima β_0-semejanza está distribuido entre todos los procesadores y su actualización se realiza entre todos.
2. La obtención de los conjuntos compactos que cambian por la llegada del nuevo documento se realiza entre todos los procesadores. La formación de *GruposAProcesarGlobal*, la distribución de *ArcosPerdidos* y la recogida de los arcos del nuevo documento se realizan en paralelo. La formación de *GruposAUnirGlobal* y los conjuntos compactos se realizan en paralelo.

IV. RESULTADOS EXPERIMENTALES

El algoritmo propuesto se implementó utilizando el estándar de programación paralela de paso de mensajes MPI. Para la realización de los experimentos se utilizó un cluster de computadoras Pentium II con 256K de memoria cache, 300MHz y 128 Mb de RAM. Estas computadoras están enlazadas por una red myrinet conectada mediante switcher. El sistema operativo utilizado fue Linux y la versión de MPI aprovecha las características de la red.

Para llevar a cabo el estudio experimental del algoritmo se tomaron 9365 noticias del periódico "El País" correspondientes a los meses de junio a diciembre de 1999. Cada noticia se representó mediante las dos componentes explicadas en la sección II. Para obtener los lemas de los términos se utilizó el analizador morfosintáctico MACO+ [13] desarrollado por la UPC.

Con esta colección de noticias se ejecutaron las variantes secuencial y paralela del algoritmo, variando la cantidad de procesadores entre 2 y 28. Los experimentos se realizaron para cantidades de noticias de 2000, 4000, 6000, 8000 y 9635. La ventana temporal establecida para la función de semejanza fue de 7 días. Además se utilizó un valor de $\beta_0 = 0{,}25$ optimizado en [9] para las noticias internacionales de junio de 1999.

En la figura 2 se muestran los tiempos obtenidos por el algoritmo paralelo al variar el tamaño de los conjuntos de noticias para 1, 2, 8 y 24 procesadores. Como puede verse, el algoritmo secuencial tiene un comportamiento cuasilineal. A medida que aumenta la cantidad de procesadores, el algoritmo tiene un comportamiento más lineal aún. Cuanto mayor es la cantidad de procesadores menores son los tiempos de cómputo.

La figura 3 muestra las aceleraciones obtenidas. Los valores de aceleración no son cercanos al ideal, lo que es de esperar en un algoritmo de coste casi lineal. Otro resultado interesante es la poca dependencia entre el tamaño de la colección de noticias y la aceleración obtenida. Esto se debe a las características de la función de semejanza entre documentos, la cual establece una ventana temporal de 7 días, por lo que las nuevas noticias prácticamente no modifi-

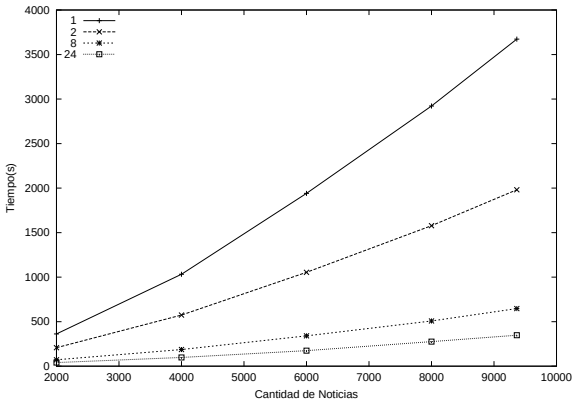

Fig. 2. Tiempos obtenidos al variar la cantidad de noticias.

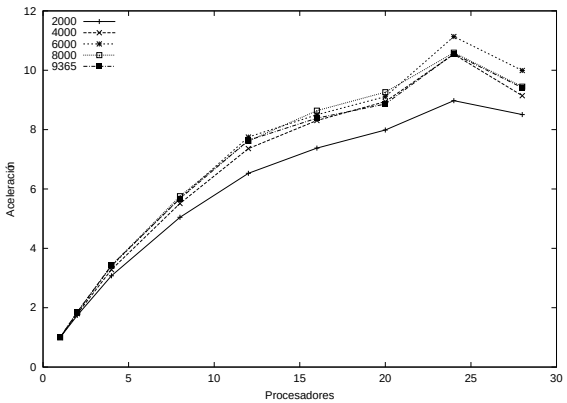

Fig. 3. Aceleraciones obtenidas al variar la cantidad de procesadores.

can los conjuntos compactos formados por noticias alejadas temporalmente de ellas.

Como resultado de la experimentación se obtuvieron 3983 sucesos, de ellos 2271 sucesos unitarios (formados por noticias aisladas) y 1712 sucesos de tamaños entre 2 y 136 noticias. La mayoría de los sucesos están conformados por menos de 20 noticias. Algunos ejemplos de sucesos detectados son: las elecciones generales en España en junio de 1999 con 34 noticias de la sección Nacionales del periódico, la guerra del filete entre Francia y Gran Bretaña relacionada con la enfermedad de las vacas locas, en octubre, con 10 noticias pertenecientes a la sección Salud, los esfuerzos realizados en los primeros días de mayo para detener los bombardeos en la guerra en Kosovo, con 24 noticias de la sección Internacionales y el Festival de Cine de San Sebastián efectuado en el mes de septiembre, conformado por 17 noticias de la sección Cultura.

V. Conclusiones

En el trabajo se expone la paralelización del algoritmo de agrupamiento compacto incremental y su aplicación a la detección de sucesos en un flujo de noticias en línea. Como resultado puede concluirse que se ha obtenido un algoritmo paralelo portable, basado en librerías de libre distribución. El mismo es eficiente y escalable con el número de procesadores si el tamaño del problema es lo suficiente grande.

Con la aplicación del algoritmo compacto incremental paralelo al problema de la deteción de sucesos se logra procesar grandes volúmenes de noticias y obtener reducciones significativas en los tiempos de procesamiento. Los experimentos fueron realizados en un cluster de computadoras personales, lo que demuestra que la detección de sucesos en noticias puede realizarse en un entorno como este, con muy buenas prestaciones respecto a su precio, aunque los costos de comunicaciones son altos respecto a los de los computos.

A pesar de que se está paralelizando un algoritmo de coste cuadrático con un comportamiento casi lineal en el problema de la deteción de sucesos y de la existencia de una alta dependencia entre los datos manejados por los distintos procesadores, se obtienen buenas prestaciones. Esto se debe al equilibrio de cargas y la minimización de los costes de las comunicaciones que se han logrado con la paralelización del algoritmo.

Referencias

[1] National Institute of Standards and Technology. The Topic Detection and Tracking evaluation plan, 2001. (http://morph.ldc.upenn.edu/TDT/).

[2] Sanz, I., Berlanga, R., Aramburu, M.J. "Gathering Metadata from Web-based Repositories of Historical Publications" 9th International Workshop DEXA'98, 473-478, IEEE Computer Soc. Press, 1998.

[3] Papka, R. "On-line New Event Detection, Clustering and Tracking". Ph.D. thesis, University of Massachusetts, Dpt. Computer Science, 1999.

[4] Hill, D. R. A vector clustering technique, *Samuelson (ed.), Mechanized Information Storage, Retrieval and Dissemination*, North-Holland, Amsterdam, 1968.

[5] Walls, F., Jin, H., Sista, S., Schwartz, R. "Topic Detection in Broadcast news". *Proceedings of the DARPA Broadcast News Workshop*, pp. 193-198, 1999.

[6] Carbonell, J., Yang, Y., Lafferty, J., Brown, R.D., Pierce, T., Liu, X. "CMU: Report on TDT2: Segmentation Detection and Tracking". Proc. of DARPA Broadcast News Workshop, 117-120, 1999.

[7] Cutting, D.R., Karger, D.R., Pedersen, J.O., Tuckey, J.W. "Scatter/Gather: a cluster-based approach to browsing large document collections". Proc. ACM SIGIR 1992, 318-329, 1992.

[8] Pons-Porrata, A., Ruiz-Shulcloper, J., Berlanga-Llavori, R., Santiesteban-Alganza, Y. "Un algoritmo de agrupamiento incremental para buscar particiones en datos mezclados". *In Pattern Recognition. Advances and Perspectives. Research on Computing Science, CIARP'2002.* México, pp. 265-276, 2002.

[9] Pons-Porrata, A., Berlanga-Llavori, R., Ruiz-Shulcloper, J.: "Detecting events and topics by using temporal references". *Lecture Notes in Artificial Intelligence 2527*, Springer Verlag, pp.11-20, 2002.

[10] Dhillon, I., Modha, B. A. Data Clustering Algorithm On Distributed Memory Multiprocesor, *Workshop on Large-scale Parallel KDD Systems*, pp. 245-260, 2000.

[11] Gil-García, R. , Badía-Contelles, J. "Algoritmo de Agrupamiento Paralelo GLC". *In Pattern Recognition. Advances and Perspectives. Research on Computing Science, CIARP'2002.* México, pp. 383-394, 2002.

[12] Wilkinson, B. , Allen, M. *Parallel programing: techniques and applications using networked workstations and parallel computers,* Prentice Hall, 1999.

[13] Carmona, J., et. al. "An Environment for Morphosintatic Processing of Unrestricted Spanish Text". In Proceeding of LREC'98, 1998.