Modelización BSP de Listas Invertidas Paralelas

V. Gil Costa, A. M. Printista
LIDIC - Departamento de Informática
Universidad Nacional de San Luis
San Luis, Argentina
{gvcosta,mprinti}@unsl.edu.ar

Mauricio Marín
Departamento de Computación
Universidad de Magallanes
Punta Arenas, Chile
mmarin@ona.fi.umag.cl

Resumen

Las listas invertidas son estructuras de datos frecuentemente utilizadas como índices para bases de datos textuales. Su propósito es acelerar la resolución de consultas sobre grandes colecciones de texto. Actualmente su aplicación más importante es sobre búsquedas en la Web. Para estos casos, el servidor debe ser capaz de procesar eficientemente miles de consultas provenientes de los usuarios de Internet, por unidad de tiempo. La demanda creciente de este tipo de servicios ha llevado a considerar la realización paralela de las listas invertidas.
En este trabajo mostramos la modelización teórica de dos estrategias de listas invertidas. Para ello se ha utilizado el Modelo *Bulk-Synchronous Parallel* BSP ya que proporciona una metodología bien estructurada y simple de diseño y análisis de algoritmos paralelos. El artículo finalmente analiza, para cada estrategia, la correspondencia entre la aproximación teórica y la implementación corriente realizada.

Palabras Claves: Paralelismo, Modelización, Superpaso, Listas Invertidas, Base de Datos Textuales.

1 Introducción

El propósito de analizar, diseñar e implementar nuevas estructuras para acceder a grandes bases de datos textuales es acelerar las operaciones de consultas sobre ellas. Asumiendo una colección de textos compuesta de un gran conjunto de documentos, una lista invertida es básicamente una tabla (el vocabulario) que mantiene todas las palabras relevantes encontradas en el texto y una lista, por cada una de esas palabras, que registra todas las ocurrencias de la palabra en la colección de textos (identificando el documento y otra información necesaria para construir las respuestas a las consultas de los usuarios). Un número de estrategias han sido propuestas recientemente para la construcción de listas invertidas [1, 3, 4]. Entre ellas tenemos las denominadas listas invertidas locales y globales [3].

La arquitectura del sistema sobre la cual se diseñan los algoritmos implementados, consiste de una máquina denominada *broker*, que es la encargada de recibir las consultas provenientes de la máquina *usuario* y de distribuirlas entre las máquinas que conforman el *servidor* de la base de datos textual. Este servidor es quien debe realizar el procesamiento de las consultas, seleccionando mediante una operación de *ranking*, los mejores documentos que serán devueltos como resultado al usuario a través de la máquina *broker*.

La indexación local es una estrategia muy simple debido a que la lista invertida se construye en base a los documentos que cada procesador posee. Cada máquina busca en dichos documentos los términos de interés, calculando la cantidad de veces que el término se repite y opcionalmente las posiciones en las que se encuentra (que es la información adicional necesaria para construir la lista invertida). La lista finalmente es ordenada lexicográficamente para facilitar la posterior resolución de consultas. Cuando la construcción de las listas invertidas ha finalizado, cada máquina del servidor posee una de estas listas, que han sido generadas considerando únicamente los documentos locales. Por lo tanto, se puede observar que cada una de las máquinas contendrá una tabla con los mismos T términos, pero la longitud de la lista de identificadores de documentos asociados es aproximadamente $1/P$, donde P es el número de máquinas de servidor. El procesamiento paralelo de una consulta u, consiste en seleccionar una máquina del servidor quien recibirá dicha consulta para luego realizar una operación de *broadcast*, y de este modo, los procesadores obtienen la misma consulta a resolver. Cuando todas las máquinas del servidor han concluido con su búsqueda local para la consulta recibida, la máquina *ranker*, seleccionada previamente por la máquina *broker*, realiza un *ranking* final de los documentos que han arribado desde las demás máquinas que constituyen el *servidor*, para luego enviarle el conjunto de los K mejores documentos al *broker*. Por último, la máquina *broker* al recibir el resultado enviado por la máquina *ranker*, simplemente se lo envía en forma de respuesta a la máquina *usuario*.

En la indexación global, la colección total de documentos es utilizada para producir una única tabla de vocabulario que es idéntica a la secuencial. Luego los T términos que forman la tabla global de términos, son distribuidos uniformemente sobre los P procesadores junto con sus respectivas listas de identificadores. Por lo tanto, luego de mapear los términos a los distintos procesadores, cada uno contiene aproximadamente T/P términos.
Para realizar el procesamiento paralelo de las consultas provenientes de las máquinas de los usuarios, cada término de la consulta es ruteado a un procesador del *servidor* a través de la máquina *broker*. Para cada término w perteneciente a la consulta u, se recupera la lista invertida asociada al término en el procesador correspondiente, luego su lista de identificadores asociados es enviada al procesador *ranker* definido para la consulta u y luego el proceso procede de manera similar a la indexación local.

2 Modelización BSP

Para realizar la construcción de los índices y el procesamiento de las consultas en forma paralela, nos hemos basado en el modelo de computación *Bulk-Synchronous Parallel - BSP*, propuesto en 1990 por Leslie Valiant [7]. La idea fundamental de BSP es la división entre computación y comunicación. Se define un "paso" como una operación básica que se realiza sobre los datos locales de un procesador. Todo programa BSP consiste en un conjunto de estos pasos, denominados "superpasos". En cada uno de ellos existe una fase de computación local independiente, seguida de una fase global de comunicaciones y todo superpaso finaliza con una sincronización por barrera que permite separar los diferentes superpasos. Cualquier petición de datos remotos se puede realizar durante el superpaso, pero estos datos no se podrán utilizar hasta entrar al siguiente superpaso, después de la sincronización [6, 8].

El costo del tiempo total de ejecución de un programa BSP, es la suma acumulativa de los costos de sus superpasos. El costo de cada superpaso es la suma de tres términos obtenidos en base a cuatro medidas: w, h, G y L, donde w es la cantidad máxima de computaciones reali-

zadas por cada procesador, h es la cantidad máxima de mensajes enviados/recibidos por cada procesador donde cada palabra cuesta G unidades de tiempo de ejecución, y L es el costo de la sincronización (*barrier*) entre los procesadores. El efecto de la arquitectura de las máquinas está considerado en los parámetros G y L. Se supone que la longitud media de las listas de identificadores es W. Para grandes bases de datos textuales se debería esperar que $W >> P$. También se debería esperar que las tablas de términos puedan ser almacenados completamente en sus respectivas memorias locales. En este trabajo se considera el costo desde que la máquina *broker* envía la consulta al servidor hasta el instante que esta máquina recibe los resultados desde el *servidor* BSP [3].

En las próximas dos subsecciones se procederá a modelar teóricamente del costo de computación de ambas estrategias utilizando el modelo BSP. Para ello, se supone un conjunto de P máquinas idénticas pertenecientes al *servidor*, cada una con su propia memoria principal y secundaria. La memoria secundaria es tratada igual que la red de comunicación. Es decir que se incluye un parámetro D para representar el costo promedio de acceder a la memoria secundaria. Este parámetro puede ser fácilmente obtenido utilizando programas *benchmark* disponibles en los sistemas Unix. La base de datos textual es distribuida entre las P máquinas que conforman el *servidor*. Si el índice de la base de datos puede ser almacenado completamente en las P memorias principales, se asume $D = 1$.

2.1 Indexación Local

Se describe a continuación el costo inferido bajo el modelo BSP para la ejecución de una única consulta utilizando la estrategia de listas invertidas locales. En esta estrategia, es necesario que cada procesador trabaje sobre el mismo conjunto de términos. Por lo tanto será necesario realizar un *broadcast* de los q términos de cada consulta, para que todos los procesadores reciban una copia. Trabajando con sólo una máquina *broker* y utilizando el método descripto en la sección anterior, se tiene que la máquina *víctima* del servidor, seleccionada por el *broker*, recuperará q términos de la consulta recibida desde su cola de entrada (con costo q), y realizará un *broadcast* de ellos a todos los demás procesadores, lo cual genera qP mensajes. Por lo tanto el costo del primer superpaso (t_1) queda determinado por el máximo de las siguientes expresiones:

$$t_{1,víctima} = \overbrace{q}^{cómputo} + \underbrace{qPG}_{comunicación} + \overbrace{L}^{sincronización}$$

$$t_{1,i} = 0 \qquad i = 0, .., P-1 \wedge i \neq víctima$$

Donde L es el costo de la sincronización *barrier*, qP es la h-relación, es decir el máximo número de mensajes enviados/recibidos en este superpaso y G es el costo en palabras de enviar el mensaje.
Luego de este *broadcast*, cada procesador del *servidor* recuperará q términos de su cola de mensajes de entrada, y trabajará en la determinación de las listas de identificadores de documentos para cada término recuperado. Si el tamaño promedio de la lista a generar es W/P, el costo de armar esta lista será: $q(W/P)D$. Resta que los procesadores envíen sus sublistas (de tamaño promedio W/P) al procesador *ranker*, para que pueda construir la lista final. El costo de este segundo superpaso es:

$$t_{2,preranker} = \overbrace{q + q(W/P)D}^{cómputo} + \underbrace{(W/P)G}_{comunicación} + \overbrace{L}^{sincronización} \qquad preranker = 0, .., P-1$$

Finalmente, en el último superpaso, la máquina *ranker* recuperará de su cola de mensajes de entrada P sublistas de tamaño W/P, construirá la lista final de tamaño W y se la enviará a la máquina *broker*. Esto insume un costo de:

$$t_{3,ranker} = P(W/P) + W + WG + L = W + W + WG + L = 2W + WG + L$$
$$t_{3,i} = 0, \qquad i = 0, .., P-1 \wedge i \neq ranker$$

Por lo tanto el costo total del procesamiento para q términos de una consulta se obtiene por la suma de los costos individuales de los tres superpasos:

$$T_3 = q + qPG + L + q + q(W/P)D + (W/P)G + L + 2W + WG + L$$
$$T_3 = 2q + W(qD/P + 2) + (qP + W/P + W)G + 3L$$

Con el mismo razonamiento, a continuación se realiza el análisis para el procesamiento de un lote de s consultas que serán suministradas al $servidorBSP$ en forma superpuesta.
En el primer superpaso, la máquina *víctima* seleccionada por el *broker*, recuperará q términos de su cola de mensajes de entrada (costo q) y realizará un *broadcast* de los mismos. Por lo tanto el costo de este superpaso es igual al costo del primer superpaso del caso analizado anteriormente:

$$t_{1,víctima} = q + qPG + L$$
$$t_{1,i} = 0, \qquad i = 0, .., P-1 \wedge i \neq víctima$$

Luego, en el siguiente superpaso, todas las máquinas del *servidor* BSP recuperarán los q términos de la primera consulta, desde su cola de mensajes de entrada. A partir de aquí, cada procesador estará habilitado para proceder con el *preranking* y el envío de su sublista de tamaño promedio W/P al *ranker*.
Además de esto, hay que tener en cuenta que uno de los procesadores del *servidor*, deberá ser escogido por el *broker* como máquina *víctima* de la segunda consulta del lote. Por lo que en su cola de mensajes, también en este superpaso, dicha máquina poseerá q términos para distribuirlos a los demás procesadores del servidor. Por lo que el costo del segundo superpaso es el máximo de las dos expresiones siguientes:

$$t_{2,víctima} = q + q(W/P)D + (W/P)G + \overbrace{q + qPG}^{broadcast} + L$$

$$t_{2,preranker} = \overbrace{q + q(W/P)D}^{preranking} + (W/P)G + L, \quad preranker = 0, .., P-1 \wedge preranker \neq víctima$$

El trabajo realizado por el servidor en el tercer superpaso es, en parte, el mismo que el realizado en el segundo superpaso. El servidor recuperará los términos de la consulta "2" y se concentrará en el proceso de *preranking*. Además, una de estas máquinas (la máquina *víctima*) deberá distribuir los q términos de la consulta "3". Adicionalmente, en este superpaso, una de las máquinas deberá realizar el proceso de *ranking*. Para ello, recuperará desde su cola de mensajes de entrada $P(W/P)$ sublistas generadas para la consulta "1" y elaborará el resultado final de tamaño W para enviársela a la máquina *broker*. Por lo tanto, el costo del tercer superpaso es el máximo de las siguientes tres expresiones:

$$t_{3,ranker} = q + q(W/P)D + (W/P)G + \overbrace{P(W/P) + W}^{ranking} + WG + L$$

$$t_{3,víctima} = q + q(W/P)D + (W/P)G + q + qPG + L$$

$$t_{3,preranker} = q + q(W/P)D + (W/P)G + L, \quad preranker = 0,..,P-1 \wedge preranker \neq víctima \wedge preranker \neq ranker$$

En este caso hay que tener en cuenta que habrá varias actividades realizándose en paralelo, por lo cual el costo insumido por un procesador en un superpaso es dependiente de la operación que oportunamente le toque ejecutar.
A partir de aquí, se puede pensar que el *servidor* entra en un estado de régimen permanente de trabajo que se mantiene mientras reciba un suministro continuo de consultas desde el *broker*. Bajo este supuesto, el costo de cualquier superpaso siguiente es idéntico al costo del tercer superpaso.
Los procesadores del *servidor*, mantendrán este estado hasta el superpaso s inclusive, donde uno de los procesadores realizará el *ranking* de la consulta "s-2", el *servidor* completo deberá realizar un *preranking* de la consulta "s-1" y uno de los procesadores realizará la distribución de la última consulta, la consulta "s". Por lo tanto el costo de este superpaso es el máximo de las tres siguientes expresiones:

$$t_{s,ranker} = q + q(W/P)D + 2W + (W/P + W)G + L$$

$$t_{s,víctima} = 2q + q(W/P)D + (W/P + qP)G + L$$

$$t_{s,preranker} = q + q(W/P)D + (W/P)G + L, \quad preranker = 0,..,P-1 \wedge preranker \neq víctima \wedge preranker \neq ranker$$

En el superpaso $s+1$, no se recibirán más consultas desde la máquina *broker*. La máquina *ranker* del *servidor* recuperará desde su cola de mensajes de entrada, las $P(W/P)$ sublistas generadas para la consulta "s-1". Además, todas las máquinas del *servidor* recuperarán los q términos de la consulta "s", harán el proceso de *preranking* y se comunicacrán con el *ranker* para enviarles sus sublistas. Por lo tanto el costo de este superpaso es el máximo de:

$$t_{s+1,preranker} = q + q(W/P)D + (W/P)G + L, \quad preranker = 0,..,P-1 \wedge preranker \neq ranker$$

$$t_{s+1,ranker} = q + q(W/P)D + 2W + (W/P + W)G + L$$

En el último superpaso, el superpaso $s+1$, sólo trabaja la máquina *ranker* seleccionada para la consulta s, y todos los demás procesadores del *servidor* permanecen ociosos. La máquina *ranker* del servidor recuperará desde su cola de mensajes de entrada, las $P(W/P)$ sublistas generadas para la consulta "s" y generará el resultado de tamaño W a ser enviado a la máquina *broker*. El costo del superpaso $s+2$ es el máximo de las siguientes dos expresiones:

$$t_{s+2,ranker} = 2W + WG + L$$

$$t_{s+2,i} = 0, \quad i = 0,..,P-1 \wedge i \neq ranker$$

Concluyendo el análisis realizado para la resolución paralela de un lote de s consultas, el costo en el *servidor BSP* propuesto es el siguiente:

$$T_{s+2} = \sum_{s=1}^{s+2} t_s$$

Si la cantidad de consultas es suficientemente grande, el costo de los dos primeros y dos últimos superpasos es despreciable en la sumatoria total.

2.2 Indexación Global

En esta sección se realizará un análisis teórico del costo de la estrategia de listas invertidas globales, para el procesamiento de una única consulta y para el procesamiento de un lote de s consultas.
El costo para el procesamiento de una única consulta utilizando la estrategia de indexación global es el siguiente. En el primer superpaso, cada procesador seleccionado por la máquina *broker* recuperará t términos (aproximadamente q/P) desde su cola de mensajes de entrada y construirá en paralelo la lista de identificadores con un costo de $t(W/P)D$, para enviársela a la máquina *ranker*. Para ello utilizará su arreglo de *pivots* armado durante la construcción de la lista invertida. Por lo tanto, el costo de este superpaso es el máximo de las siguientes expresiones:

$$t_{1,prerankers} = \overbrace{t + t(W/P)D}^{cómputo} + \underbrace{(W/P)G}_{comunicación} + \overbrace{L}^{sincronización}$$

$$t_{1,i} = 0, \quad i = 0, .., P-1 \wedge i \neq prerankers$$

Donde L es el costo de la sincronización *barrier* y (W/P) es la h relación, es decir la máxima cantidad de mensajes de entrada/salida para este superpaso.
Luego, en el segundo superpaso, la máquina *ranker* recupera P sublistas de tamaño promedio (W/P) de su cola de mensajes de entrada y construye el resultado final de tamaño W, para enviárselo a la máquina *broker*. Por lo tanto, el costo de este superpaso es el máximo de:

$$t_{2,ranker} = P(W/P) + W + WG + L = 2W + WG + L$$

$$t_{2,i} = 0, \qquad i = 0, .., P-1 \wedge i \neq ranker$$

Finalmente, el costo total para esta estrategia es la suma de los costo de los dos superpasos descriptos anteriormente:

$$T_2 = t + t(W/P)D + (W/P)G + L + 2W + WG + L$$
$$T_2 = t(1 + (W/P)D) + 2W + WG(1/P + 1) + 2L$$

Con el mismo razonamiento, a continuación se realiza el análisis para el procesamiento de un lote de s consultas que serán suministradas al $servidorBSP$ en forma superpuesta.

En el primer superpaso, las máquinas del *servidor* seleccionadas por la máquina *broker* utilizando el arreglo de *pivots*, recuperarán t términos desde sus colas de mensajes de entrada y realizarán un *preranking* para obtener las sublistas de tamaño promedio W/P a ser enviadas a la máquina *ranker*. El costo de este superpaso es el máximo de:

$$t_{1,prerankers} = t + t(W/P)D + (W/P)G + L \quad \text{(Para toda máquina seleccionada)}$$

$$t_{1,i} = 0, \quad i = 0, .., P-1 \wedge i \neq prerankers$$

En el segundo superpaso, las máquinas determinadas los por los *pivots* (*prerankers*) recuperarán desde su cola de mensajes de entrada la segunda consulta a ser procesada, con el mismo costo descripto para el superpaso anterior. Además, otra máquina del servidor, la máquina *ranker* recuperará $P(W/P)$ sublistas generadas para la consulta "1" por todos los procesadores del servidor en el superpaso anterior, para obtener la lista final y enviársela al *broker*. Por lo tanto, se tiene un costo de:

$$t_{2,prerankers} = t + t(W/P)D + (W/P)G + L, \quad prerankers = 0, .., P-1 \wedge prerankers \neq ranker$$

$$t_{2,ranker} = 2W + WG + L, \qquad ranker \neq prerankers$$

$$t_{2,i} = 0, \qquad i = 0, .., P-1 \wedge i \neq prerankers \wedge i \neq ranker$$

Observar que es posible que la máquina *ranker* también sea una máquina receptora, debido a que en esta estrategia puede existir más de una máquina *preranker*. En este caso el costo para esta máquina es el siguiente:

$$t_{2,j} = t + t(W/P)D + (W/P)G + 2W + WG + L, \quad j = preranker \wedge j = ranker$$

Como se puede observar, a partir de este momento el trabajo del servidor se estabiliza. Es decir que desde el segundo superpaso hasta que el servidor recibe la última consulta, el trabajo realizado por el servidor será una sucesión de superpasos que consitirán del *preranking* y del *ranking* de las consultas.
Durante el superpaso s, las máquinas *prerankers* recuperarán los t términos de la última consulta a ser procesada con un costo t y construirán las sublistas que serán enviadas a la máquina *ranker* con un costo de $t(W/P)D$. También otra máquina del servidor, la máquina *ranker*, recuperará las $P(W/P)$ sublistas y construirá la lista de tamaño W a ser enviada como respuesta a la consulta "s-1" a la máquina *broker*. El costo de este superpaso es el máximo de las siguientes expresiones:

$$t_{s,prerankers} = t + t(W/P)D + (W/P)G + L, \quad prerankers = 0, .., P-1 \wedge prerankers \neq ranker$$

$$t_{s,ranker} = 2W + WG + L, \quad ranker \neq prerankers$$

$$t_{s,i} = 0, \qquad i = 0, .., P-1 \wedge i \neq rankers \wedge i \neq preranker$$

Recordar que cuando la máquina *ranker* es también escogida com *víctima*, el tiempo de procesamiento insumido por la misma es:

$$t_{s,j} = t + t(W/P)D + (W/P)G + 2W + WG + L, \qquad j = 0, .., P-1 \wedge j = prerankers \wedge j = ranker$$

Por último, en el superpaso $s+1$, ya no se reciben más consultas desde el *broker* y sólo queda construir la lista resultado de tamaño W. Es decir que la única máquina que trabaja en este superpaso es la máquina *ranker* y todas las demás permanecen ociosas.

$$t_{s+1,ranker} = 2W + WG + L$$

$$t_{s+1,i} = 0, \quad i = 0, .., P-1 \wedge i \neq ranker$$

Por lo tanto, el costo total para la resolución paralela de un lote de "s" consultas utilizando el modelo de computación BSP es el siguiente:

$$T_{s+1} = \sum_{s=1}^{s+1} t_s$$

3 Análisis Teórico versus Análisis Empírico

En esta sección se realiza una exhaustiva discusión respecto a la convergencia entre el modelo teórico que sustentó la fase de diseño y los resultados obtenidos de una implementación real, la cual es descripta en [2].

Para realizar un análisis empírico se han ejecutado distintos casos de pruebas, tanto sobre una base de datos textual uniforme, donde todos los términos de las consultas tienen la misma probabilidad de aparecer, y sobre una base de datos no uniforme, donde en los términos de las consultas aparecen las cuatro letras del idioma español más frecuentes (a, c, m, p).
Se han llevado a cabo experimentos utilizando las estrategias de indexación global y local sobre un cluster de *12 SMP duales*, conectadas mediante una red *Fast Ethernet*. Para cada situación se ha modelado analíticamente ambas estrategias de indexación y se ha realizado un análisis experimental en base a la implementación corriente, analizando si existe una coherencia entre la inferencia teórica BSP y los resultados reales obtenidos.

3.1 Estrategia de Indexación Local - Análisis de un lote de consultas

Reveamos el modelo BSP resultante de esta estrategia:

$$T(IndL, P, S) = \sum_{s=1}^{S} (t(IndL, P, s))$$

donde:

$$t(IndL, P, s) = \max(t_{s,ranker}, t_{s,prerankers}, t_{s,victima})$$

es decir:

$$t(IndL, P, s) = \max_{s} \left\{ \begin{array}{l} q + q(W/P)D + 2W, \\ \\ q + q(W/P)D, \\ \\ 2q + q(W/P)D \end{array} \right\} + h_s * G + L_P$$

donde la $h-$relación de cada superpaso es:

$$h_s = ((W/P)P + W + qP)) = 2W + qP$$

Recordemos que en esta estrategia todos los procesadores realizan una operación de *preranking* sobre los q términos recibidos. Denominamos *preranking* a la acción completa de leer de la cola de entrada q términos, armar la sublista invertida y comunicarla al *ranker*. Su costo está expresado en términos de W/P, lo que señala que al incrementar P el tiempo del *preranking* tiende a disminuir. Además, ya en régimen estable, en cada superpaso, una de las máquinas del *servidor paralelo BSP* será escogida como *víctima* y una de ellas como *ranker*.
Esto produce que en cada superpaso, el máximo tiempo al que se sincronizan los procesadores sea al costo de cómputo de la máquina *ranker*, mas el tiempo de la $h-$relación y ambos están limitados inferiormente en $2W$. Además, a medida que crece P disminuye el tiempo de cómputo (qW/P) y aumenta el de comunicación (qP), ya que este último está determinado fundamentalmente por el costo de la operación *broadcast*. De lo anterior, el costo de realizar la ejecución de un lote de consultas utilizando listas invertidas locales, depende tanto del tamaño de la lista como del *servidor BSP*.

Con valores de q adecuados, aproximadamente 10 términos por consultas, el modelo predice una aceleración importante, pero la misma disminuye al aumentar P.
Las Figuras 1 y 2, muestran la performance real del algoritmo en una plataforma de hasta diez procesadores, trabajando con una base de datos uniforme (uni) y una base de datos no uniforme (4-letras), para un lote de 100 y 1000 consultas. Como se puede observar en la Figura 2 para $P > 5$, el *speedup* comienza a decrecer, coincidiendo con la inferencia teórica respecto a que los tiempos de comunicación y sincronización obtenidos tendrían una influencia directa sobre la aceleración que se puede lograr.

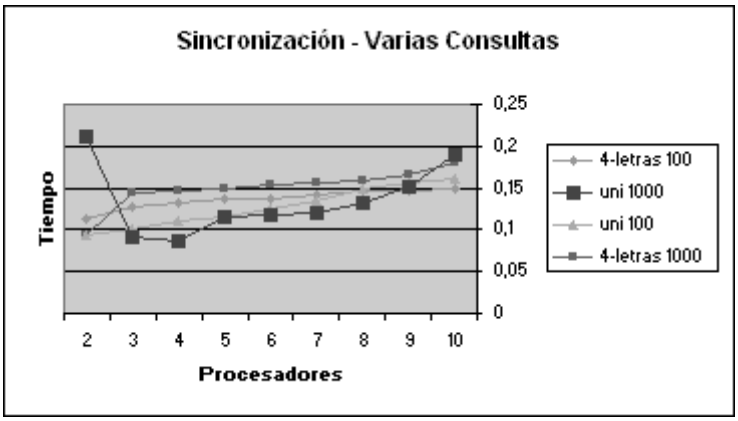

Figura 1: Tiempos de Sincronizacion -Estrategia de Lista Invertida Local

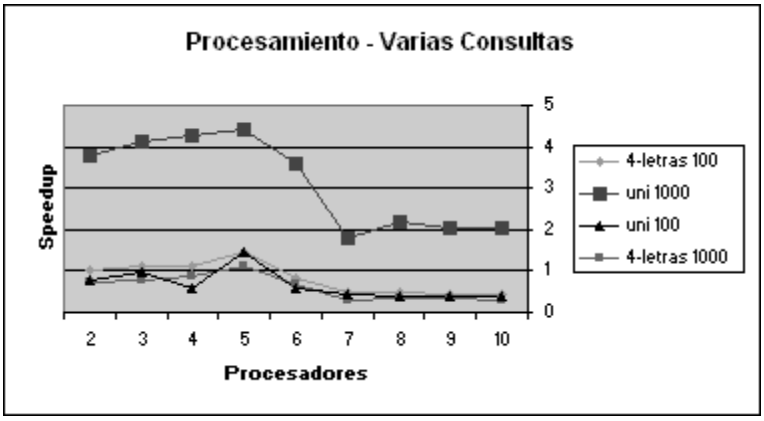

Figura 2: Speedup - Estrategia de Lista Invertida Local

3.2 Estrategia de Indexación Global- Análisis de un lote de consultas

Analicemos el modelo analítico resultante de esta estrategia:

$$T(IndG, P, S) = \sum_{s=1}^{S} (t(IndL, P, s))$$

donde:

$$t(IndG, P, s) = \max(t_{s,prerankers}, t_{s,ranker}, t_{s,pre+rankers})$$

es decir:

$$t(IndG, P, s) = \max_s \left\{ \begin{array}{l} t + t(W/P)D, \\ 2W, \\ t + t(W/P)D + 2W \end{array} \right\} + h_s * G + L_P$$

para una h-relación: $h_s = (W/P)N_s + W$

(donde N_s es la cantidad de máquinas seleccionadas según pivots durante el superpaso s)

La $h-$relación en cada superpaso, $(W/P)N_s$, se debe a que consideramos que las entradas y salidas de datos tienen lugar en estricta secuencia (nuestra máquina BSP es un cluster), por lo que la medida de la cantidad comunicada por el procesador i es definida como la suma $h_{s,i} = in_{s,i} + out_{s,i}$ [6]. De lo anterior resulta que la $h-$relación es la misma (enviar n sublistas y una lista final) independientemente a que la máquina *ranker* coincida con una de las *prerankers*. También se puede observar que la cantidad de comunicación tiene un límite superior en $2W$ mensajes (esto es para $N = P$) y este tiempo disminuye a medida que la brecha entre P y N crece. Esto indica un aumento del desbalance de la carga de trabajo de los servidores de consultas de la base de datos textual.

En cuanto al cómputo, se puede observar que en el caso de los superpasos donde la máquina del servidor no actúa de *ranker* y de *preranker*, el cómputo del superpaso $(2W)$ es constante e independiente de P. Esta situación es mas probable cuando sea mayor el tamaño sel servidor. Cuando una misma máquina cumple las dos funciones, entonces el cómputo logra una cota superior en $3W$, el cual disminuye al aumentar P.

Es importante observar que para servidores pequeños, el tiempo de cómputo y comunicación es condiderable, ya que existe la probabilidad de que una misma máquina realice en el mismo superpaso un *preranking* y un *ranking* final; pero en estos casos el tiempo de sincronización es relativamente pequeño. Al crecer P, esta probabilidad disminuye, por lo tanto pueden haber algunos superpasos sincronizados al tiempo de la máquina *ranker* y este tiempo no depende de P. Además, en estos casos, la influencia de un costo de sincronización mayor afecta el *speedup* esperado.

La gráfica de la Figura 4 permite observar los valores de *speedup* obtenidos para un servidor de hasta diez procesadores. Llegado un punto, el tiempo de sincronización comienza a ganar sobre el tiempo de procesamiento requerido (Ver Figura 3) y el *speedup* se mantiene bajo.

4 Conclusiones

Las experiencias realizadas hasta el momento exhiben, en general y para la mayoría de los casos de prueba, un alto grado de coincidencia entre los resultados teóricos y experimentales, los cuales indican una predicción confiable del comportamiento de los algoritmos que implementan la resolución de consultas utilizando las estrategias de listas invertidas locales y listas invertidas globales.

Para los algoritmos de resolución de consultas presentados en este trabajo, utilizando tanto la estrategia de indexación local como la global, y basados en el modelo *BSP*, se ha podido

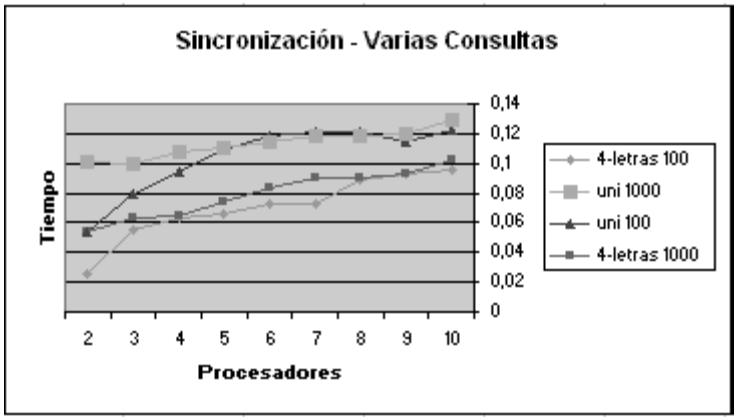

Figura 3: Tiempos de Sincronizacion -Estrategia de Lista Invertida Global

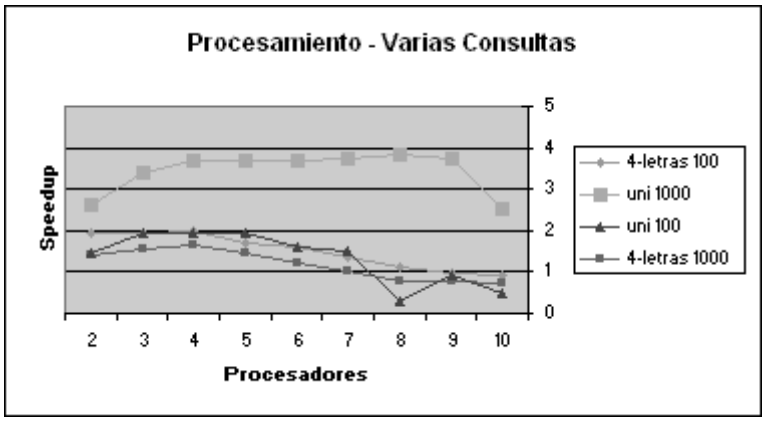

Figura 4: Speedup - Estrategia de Lista Invertida Global

observar que en sistemas con listas invertidas pequeñas, se visualiza una modesta mejora de performance de la técnica global sobre la local. Pero para sistemas con listas invertidas de gran tamaño, que es donde se justifica el uso del paralelismo, el costo de ambas estrategias tiende a ser similar.
Sin embargo queda analizar nuevas alternativas de diseño que permitan una mejora en el *speedup* alcanzado por estas estrategias.

Agradecimientos

Este trabajo ha sido llevado a cabo en cooperación entre las Universidades de San Luis (Argentina) y Magallanes (Chile). Ha sido el resultado de la iniciativa de la Red Iberoamericana de Tecnologías del Software, `RITOS2`, incluida en el subprograma `VII` de `CYTED` (`RED-VIIJ`).

Agradecemos a la Unidad de Servicio Computacional (para aplicaciones científicas) del Instituto de Matemática Aplicada de la Universidad Nacional de San Luis (IMASL), por permitirnos realizar nuestros experimentos sobre su cluster de procesadores.

Referencias

[1] A.A. MacFarlane, J.A. McCann, y S.E.Robertson.*"Parallel Search Using Inverted Files"*. In the 7th. International Symposium on String Processing and Information Retriaval,2000.

[2] G. V. Gil Costa, "Procesamiento Paralelo de Queries sobre Base de Datos Textuales". Tesis de licenciatura. Universidad Nacional de San Luis. 2003.
[3] M. Marin, C. Bonacic y S. Casas. *"Analysis of two indexing structures for text databases"*, Actas del VIII Congreso Argentino de Ciencias de la Computación (CACIC2002). Buenos Aires, Argentina, Octubre 15 - 19, 2002.
[4] M. Marin, *"Parallel text query processing using Composite Inverted Lists"*, In proceedings of the Second International Conference on Hybrid Intelligent Systems (Invited session on Web Computing), Dec. 2002 (IOS Press).
[5] Sergery Melnik,Siriam Raghavan, Bervely Yang y Hector Garcia-Molina *"Building a Distributed Full-Text Index for the Web"*. ACM Transactions on Information Systems,2001.
[6] D.B. Skillicorn, J.M.D. Hill y W.F. MacColl.*"Questions and answers about BSP"*.Reporte técnico PRG-TR-15-96, Laboratorio de Computación, Universidad de Oxford,1996.
[7] Valiant L.G. *"Bridging Model for Parallel Computation"*, Communications of the ACM, vol. 33, numero 8, pags. 103 a 111, 1990.
[8] BSP Worldwide Standart, http://www.bsp-worldwide.org/.