Implementación de un Digesto Digital Paralelo *

Esteban Gesto, Daniel Laguía, Natalia Trejo, Osiris Sofia
Universidad Nacional de la Patagonia Austral
Río Gallegos, Argentina
{egesto;dlaguia;ntrejo;osofia}@unpa.edu.ar

and

José Canumán
Universidad de Magallanes
Punta Arenas, Chile
jose.canuman@umag.cl

Resumen

El crecimiento de la cantidad de información que se pone a disposición en Internet a través de la Web presenta el desafío de satisfacer, en el menor tiempo posible, a los clientes que realizan búsquedas sobre esa información y a la vez mejorar el uso eficiente de los recursos. Los modelos de computación paralela permiten acercarse a este objetivo. Este trabajo presenta una solución eficiente y de bajo costo basada en el modelo de computación *Bulk Synchronous Parallel*, para la implementación de un Digesto Digital basado en un motor de búsquedas paralelo que utiliza bases de datos relacionales, en un entorno de acceso Web.

Palabras claves: Bases de Datos, Procesamiento Paralelo de Consultas SQL, Computación Paralela y Distribuída, BSP

1. Introducción

La web se ha convertido en un recurso ubicuo para la computación distribuida, haciendo relevante la investigación de nuevos caminos para proveer acceso eficiente a los servicios disponibles en los sitios dedicados. El crecimiento exponencial que ha experimentado desde sus comienzos en cuanto al volumen de información y al número de usuarios que la utilizan hace que la búsqueda, organización, acceso y mantenimiento de sus contenidos sea cada vez más difícil.

En respuesta a esta expansión de las fuentes potenciales de información, los motores de búsqueda han hecho énfasis en ampliar su velocidad y cobertura, brindando poca importancia a la eficiencia.

Debido a esto, diversos estudios se han abocado al desarrollo de nuevas estrategias que permitan satisfacer estas demandas a través del procesamiento paralelo, el cual ha demostrado ser un paradigma que permite mejorar los tiempos de ejecución de los algoritmos.

En este trabajo se propone una solución basada en el modelo BSP [13, 4] de computación paralela, el cual utiliza una configuración de base de datos distribuida para acelerar las consultas, realizando el procesamiento de la cola de consultas en forma secuencial.

La implementación de este sistema requirió en primer lugar la modificación de la librería de comunicaciones de la Paderborn University, *BSP-PUB* [5, 3], para hacer posible la ejecución de programas BSP a través de sockets. Esta modificación permite la ejecución como *daemon* (es decir, con la capacidad de mantenerse en ejecución en forma continua) del módulo encargado de recibir las consultas, lo cual no era posible realizar con la versión estándar de la librería [2].

La máquina en la que reside este *daemon* es denominada *broker*, y forma parte del *cluster* o red de computadoras compuesta por P procesadores. Para evitar la sobrecarga de este procesador, no realiza consultas locales sino que únicamente administra la cola de consultas recibidas desde el servidor web, las distribuye entre cada uno de los procesadores restantes, y recopila los resultados obtenidos, devolviéndolos agrupados al servidor.

Cada una de las restantes máquinas del *cluster* posee un servidor de base de datos secuencial MySQL, y trabaja localmente sobre una porción de los datos.

Un logro a destacar es que la modificación de la librería estándar de BSP-PUB nos ha permitido implementar una aplicación real y funcional,

*Este trabajo fue financiado por la Universidad Nacional de la Patagonia Austral, Santa Cruz, Argentina, Proyecto 29/A164

utilizando tecnología gratuita existente, con costos de implementación y operación muy bajos, ya que se utiliza equipamiento de bajo costo y software libre, a la vez que obtener un rendimiento cercano al óptimo en cuanto a la eficiencia en la utilización de los recursos involucrados. Adicionalmente hemos propuesto una estrategia simple para distribuir uniformemente los registros almacenados en las bases de datos locales de cada uno de los procesadores que componen el cluster.

En las secciones siguientes brindaremos un panorama del funcionamiento del modelo de computación paralela BSP, explicaremos la configuración de cada uno de los componentes mencionados, así como describiremos las responsabilidades de los mismos y finalmente presentaremos los resultados preliminares de laboratorio logrados a través de la aplicación de nuestras investigaciones en un desarrollo concreto que tiene por objetivo la publicación de los documentos producidos por los organismos de gobierno de la Universidad y que hemos denominado *Digesto Digital Institucional*.

2. Modelo de computación paralela BSP

Varios productos comerciales fueron desarrollados para máquinas multiprocesadores de memoria compartida y distribuida, y más recientemente para *clusters* de computadores. Todos estos desarrollos están basados en modelos tradicionales de computación paralela como paso de mensajes y memoria compartida. En este trabajo se presenta una solución funcional basada en un modelo relativamente nuevo de computación paralela llamado *Bulk Synchronous Parallel*, BSP.

En BSP un computador paralelo es visto como un conjunto de procesadores con memoria local e interconectados a través de una red de comunicaciones de topología transparente al usuario. En este modelo, la computación es organizada como una secuencia de *supersteps*. Tal como lo indica la Fig. 1, un superstep está formado por una fase en la que cada procesador puede realizar operaciones sobre datos locales únicamente y depositar mensajes a ser enviados a otros procesadores. Al final del superstep, todos los mensajes son enviados a sus destinos y los procesadores son sincronizados en forma de barrera para iniciar el siguiente superstep. Es decir, los mensajes están disponibles en sus destinos al instante en que se inicia el siguiente superstep.

El modelo práctico de programación paralela en BSP es el conocido SPMD (Simple Program Multiple Data), el cual es realizado mediante P copias del mismo programa corriendo en un cluster de P procesadores, cada una actuando sobre un subconjunto de los datos, donde la comunicación y sincronización de las copias es realizada mediante librerías tales como BSPlib o BSP-PUB. Enfatizamos que el modelo BSP es en realidad un paradigma de programación en paralelo y no una librería de comunicaciones en particular. En la práctica, es ciertamente posible implementar programas BSP utilizando librerías tales como *PVM (Parallel Virtual Machine)* y *MPI (Message Passing Interface)*. Nótese que varios estudios han mostrado que algoritmos BSP presentan un desempeño más eficiente que sus respectivas contrapartes en los enfoques de paso de mensaje y memoria compartida en varias aplicaciones.

La estructura del modelo BSP facilita la predicción del desempeño de programas y algoritmos. El costo de un programa está dado por la suma del costo de todos sus supersteps, donde el costo temporal de cada uno de ellos está dado por la suma del tiempo de computación sobre datos locales, el tiempo de comunicación entre procesadores y el tiempo de sincronización.

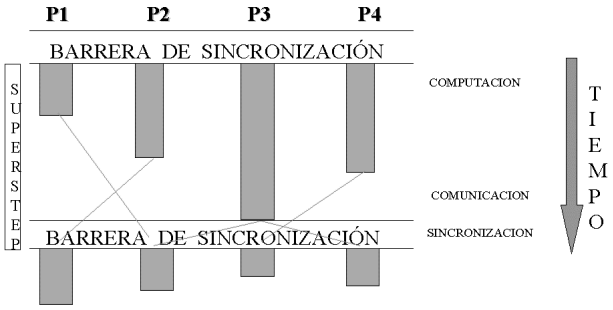

Figura 1: *Modelo BSP y supersteps.*

3. Configuración del Motor de Búsquedas Paralelo

Esta herramienta de software se compone de los siguientes elementos: una aplicación web desarrollada fundamentalmente con PHP, instalada en un servidor Web con infraestructura *LAMP* (Linux, Apache, MySQL, PHP), un servidor de base de datos distribuida implementado con MySQL, un programa ejecutable desarrollado bajo el modelo de computación paralela BSP y un *broker* realizado en base a la modificación del *daemon* del BSP-PUB, *pubd*.

Estos componentes se interrelacionan como se explica a continuación: en primer lugar los clientes acceden a la aplicación a través del servidor web, donde elaboran la consulta a realizar. El servidor web envía la consulta al *broker* a través de sockets, ejecutando el programa BSP en cada procesador que forma parte del *cluster*. En cada

procesador se ejecuta la consulta en el servidor local de base de datos MySQL y, cuando se produce la etapa de sincronización del *superstep* BSP, los resultados son enviados a la máquina *broker*, quien los agrupa y envía al servidor Web a través de los sockets definidos para que de esta manera el cliente pueda obtener los resultados requeridos.

A continuación se describen los componentes mencionados:

3.1. Configuración del Servidor Web

Para este trabajo se ha desarrollado una aplicación Web donde los clientes pueden realizar consultas relativas a los documentos emanados por las autoridades de Gobierno de la Universidad. Adicionalmente, el servidor Web almacena un archivo en formato PDF por cada uno de los documentos almacenados en la Base de Datos. Es este archivo PDF el que será accedido por los clientes una vez que reciban las tuplas que coincidan con el criterio de búsqueda ingresado. Cada tupla cuenta con un *link* al archivo PDF correspondiente. Se ha tomado esta decisión para minimizar el tráfico interno dentro del *cluster*, ya que cada cliente accederá a un archivo PDF por vez, por lo que el procesamiento paralelo para esta etapa no aportaría beneficios adicionales.

3.2. Configuración del Servidor de Base de Datos Distribuida

La implementación del servidor es como sigue. Existe un conjunto P de procesadores ejecutando los superstep de BSP. En cada máquina existe un administrador de bases de datos relacionales (MySQL en nuestro caso). Este administrador es operado mediante instrucciones en lenguaje SQL enviadas a través de una conexión *socket* implementada por una API para C++. De esta manera, cada máquina del cluster puede ejecutar comandos SQL sobre su base de datos local.

Cada una de las P máquinas mantiene el mismo esquema de la base de datos, es decir, las mismas tablas pero con distintas tuplas. Las tuplas están distribuidas uniformemente en la base de datos y almacenan la información como texto plano para la realización de las consultas.

Si bien se consideraron varias alternativas para realizar esta distribución, la solución adoptada consiste en realizar una consulta a todos los procesadores del cluster sobre la cantidad de registros existentes en cada uno de ellos antes de realizar una inserción. Para realizar esta operación se seleccionará siempre el procesador con el número mínimo de registros existentes. De esta manera es posible administrar en forma automática la distribución uniforme de la base de datos ante eliminación o inserción de registros e incluso ante cambios en la cantidad de procesadores que componen el cluster.

3.3. Broker o Agente de Gestión

La gestión de las consultas es crítica para un aprovechamiento global de los recursos, y también para garantizar que la arquitectura es adecuadamente modular y escalable. En el modelo BSP, esta optimización depende del tipo de consultas, generalmente determinada por la programación de un *broker* o agente de gestión.

Este *broker* tiene por responsabilidad administrar una cola de consultas recibidas desde la máquina *front-end*, ya que cada consulta se realiza en forma secuencial con respecto a las otras consultas. Adicionalmente realiza la tarea de intermediación entre la máquina *front-end* y el *cluster*.

Al investigar la posibilidad de implementar un sistema operacional que recibiera consultas de los clientes a través del servidor Web, nos encontramos con la imposibilidad de utilizar la librería BSP-PUB tal como fue desarrollada, ya que la misma prevee la ejecución de los programas a través de una ventana de comandos que se carga al ejecutar el *daemon* que genera la intercomunicación de los procesadores participantes y genera los buffers que se utilizarán en esa comunicación. Este *daemon* está definido como *pubd*. Por esta razón fue necesario modificar el *daemon* pubd para que permitiera la ejecución de programas BSP a través de *sockets*, permitiendo el acceso al mismo directamente desde la aplicación Web desarrollada.

Es importante destacar que si bien el *broker* participa del cluster BSP, este tiene un tratamiento especial, no contando con base de datos, y por lo tanto no se realizan búsquedas en ese procesador. Esto se ha definido así ya que en la misma máquina del *broker* se encuentra el servidor Web y, en cualquier caso, la existencia dentro del cluster de un *broker* participando además como servidor de base de datos generaría una carga adicional que haría perder el balance del sistema, afectando la performance global y convirtiéndose en un posible cuello de botella. Alternativamente el broker podría ubicarse en una máquina distinta al servidor web.

Al culminar el procesamiento de las consultas, los procesadores involucrados devuelven a *P0* (el procesador que llamó a la ejecución del programa BSP, en este caso el que contiene pubd en ejecución, es decir el *broker*) los resultados obtenidos, y éste los almacena en un archivo que será consultado por el servidor web, de manera de poder entregar al cliente que realizó la solicitud los resultados en forma parcial, de acuerdo a la demanda del usuario.

Para evitar que el *broker* se transforme en un cuello de botella del sistema se ha optado por realizar el almacenamiento de las consultas en un archivo temporal en el broker, el cual podrá ser consultado por la sesión iniciada por el cliente, y que se eliminará cuando se cierre dicha sesión.

4. Digesto Digital Institucional

La configuración del motor de búsqueda expuesta precedentemente se ha implementado exitosamente en una aplicación en etapa de prueba que hemos denominado Digesto Digital Institucional. Originalmente el término *Digesto* se aplicó a la codificación del Derecho Romano, pero actualmente y por extensión se conoce como digesto a la compilación ordenada de toda norma jurídica. El Digesto Institucional permite acceder a todo lo actuado, sancionado y legislado en el tiempo, por una Institución dada. Constituye el cuerpo de leyes o reglamentaciones por el cual se rige la actuación y las decisiones de una administración, compendiando además, todo lo resuelto o actuado en función y con atención a ese conjunto de reglamentaciones básicas.

Hemos optado por implementar el Digesto Digital de la Universidad Nacional de la Patagonia Austral, ya que el volumen de datos involucrados resulta atractivo para realizar las pruebas de laboratorio de esta investigación. En principio hemos estimado un volumen de datos de al menos 500.000 páginas de documentos correspondientes a los órganos de gobierno de la Universidad de los últimos 10 años, las que se almacenarán en formato texto en la base de datos distribuida (lo que permitirá realizar las búsquedas) y en formato de archivos PDF almacenados en el servidor web (para poder obtener un copia con autenticación por parte de la Universidad).

4.1. Resultados Preliminares

Luego de realizar en investigaciones anteriores [8, 6, 7, 9, 10, 12, 11, 1] diversas consideraciones con respecto a servidores paralelos sobre bases de datos distribuidas a través del modelo de computación paralela BSP, hemos estimado conveniente aplicar los resultados de laboratorio obtenidos, fundamentalmente a través de simulaciones, en una sistema real que nos permita por una parte verificar los resultados obtenidos en laboratorio y por otro permitir el desarrollo de una aplicación que pueda ser ampliada y mejorada a través de los aportes de los usuarios e investigadores que accedan a la misma.

Resultados preliminares de las pruebas realizadas con el compendio de normas almacenadas hasta el momento demuestran valores similares a los obtenidos en anteriores pruebas de laboratorio. En este caso sólo fue necesario realizar la simulación de las consultas generadas por los clientes. El desempeño paralelo, en comparación al secuencial, obtiene mejor rendimiento cuanto más tuplas existen en las tablas, ya que se realiza un procesamiento local más intensivo. En la figura 2 se muestra la relación tiempo_secuencial/tiempo_paralelo para el caso de un cluster compuesto por 4 procesadores y un promedio de resultados de 1000 tuplas para cada consulta. Como puede observarse, a medida que aumenta la cantidad de tuplas en la base de datos la relación indicada se acerca al óptimo, que para este caso es 4. Debe considerarse que para el caso de pocos documentos almacenados, la mejora en el procesamiento local no es suficiente para contrarrestar los costos de comunicación para la transmisión de las 1000 tuplas resultantes.

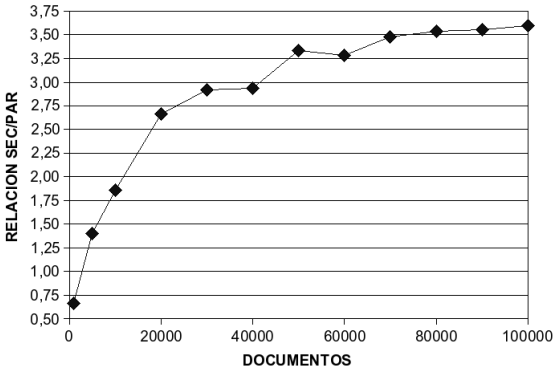

Figura 2: *Relación entre el tiempo secuencial/paralelo de acuerdo al número de documentos almacenados en la base de datos para un promedio de 1000 tuplas de resultados para una consulta.*

Un aspecto interesante es que el efecto del aumento en comunicación en el caso paralelo no es muy significativo frente a la mayor actividad de disco generada producto de tablas de mayor tamaño en el caso secuencial.

4.2. Trabajos Futuros

Trabajos futuros permitirán mejorar la administración de la cola de consultas manejada por la máquina broker, por ejemplo a través de la evaluación de prioridades. Otro tema de estudio es la recuperación de la base de datos distribuida a raíz de la caída de una máquina del cluster, mediante la replicación de registros u otra estrategia. También es posible su aplicación a redes de Datos del tipo *Grid*, si se piensa que en una aplicación como el *Digesto* pueda plantearse la necesidad de contar con una base de datos ditribuida geográficamente para respetar la autonomía de las organismos intervinientes. Por último es posible investigar la

aplicación de este modelo en la conexión de las Bases de Datos relacionales con una ontología legal y el acceso a la misma a través del desarrollo de una web semántica.

5. Conclusiones

En este trabajo hemos presentado una solución concreta al problema de acceso a grandes volúmenes de datos a través de la Web, mediante el desarrollo de varios componentes que conforman un motor de búsquedas paralelo, con acceso a una base de datos distribuida, implementando el modelo de computación paralela BSP, en particular a través de la librería BSP-PUB. Este modelo soporta una metodología estructurada de diseño de software que es simple de utilizar y permite el uso de tecnología existente y gratuita para obtener sistemas de bajo costo y alta eficiencia.

En particular, en este trabajo nos hemos centrado en el desarrollo del servidor de consultas y el broker asociado, que permite distribuir las consultas y enviar los resultados a los clientes que las han generado. Con las modificaciones realizadas a la librería BSP-PUB ahora es posible su utilización en sistemas de búsqueda y recuperación de información en un entorno paralelo con bases de datos relacionales, permitiendo lograr eficiencia a un bajo costo, con tiempos de respuesta superiores a los sistemas secuenciales.

En los resultados preliminares de laboratorio obtenidos a través de un sistema denominado Digesto Digital Institucional, se han obtenido valores cercanos al óptimo en cuanto a eficiencia en relación con el caso secuencial, para el caso de grandes cantidad de registros, donde se justifica la utilización de modelos paralelos.

Referencias

[1] Patricia Barani, Esteban Gesto, Daniel Laguía, Albert Sofía, José Canumán, and Mauricio Marín. Parallel digital digest on bsp. *REVISTA AUSTRO INGENIERÍA, Facultad de Ingeniería, UMAG*, 9(1):36–42, 2005.

[2] O. Bonorden, B. Juurlink, I. von Otte, and I. Rieping. The paderborn university bsp (pub) library - design, implementation and performance. In *13th International Parallel Processing Symposium*, San Juan, Puerto Rico, Apr. 1999.

[3] O. Bonorden, B. Juurlink, I. von Otte, and I. Rieping. The paderborn university bsp (pub) library. *Parallel Computing*, 29(2):187–207, Feb. 2003.

[4] BSP and Worldwide Standard. http://www.bsp-worldwide.org/.

[5] BSP PUB Library at Paderborn University. http://www.uni-paderborn.de/bsp.

[6] M. Marín, J. Canuman, M. Becerra, D. Laguia, and O. Sofia. Procesamiento paralelo de consultas sql generadas desde la web. In *Jornadas Chilenas de Computación 2001*, Punta Arenas-Chile, Nov. 2001.

[7] M. Marín, J. Canuman, M. Becerra, D. Laguia, and O. Sofia. Servidor paralelo sql-bsp para aplicaciones web. In *VII Congreso Argentino de Ciencia de la Computación*, El Calafate - Argentina, Oct. 2001. CACIC 2001.

[8] M. Marin, J. Canumán, and D. Laguia. Un modelo de predicción de desempeño para bases de datos relacionales paralelas sobre bsp. In *VI Congreso Argentino de Ciencia de la Computación*, Ushuaia - Argentina, Oct 2000. CACIC 2000.

[9] M. Marín and S. Casas. On bulk-synchronous distributed-memory parallel processing of relational-database transactions. In *VII Congreso Argentino de Ciencia de la Computación*, El Calafate - Argentina, Oct. 2001. CACIC 2001.

[10] M. Marín and S. Casas. Procesamiento paralelo de consultas a bases de datos textuales distribuidas. In *III Workshop de Investigadores en Ciencias de la Computación*. WICC 2001, May. 2002.

[11] Paula Millado, Daniel Laguia, Albert Sofia, and Mauricio Marín. Representación visual para la administración del procesamiento paralelo de consultas sql. In *IX Congreso Argentino de Ciencia de la Computación*, La Plata - Argentina, Oct. 2003. CACIC 2003.

[12] Paula Millado, Daniel Laguia, Albert Sofia, and Mauricio Marín. Visualización gráfica de consultas sql en paralelo. In RITOS2, editor, *Ingenierá de Software en la década del 2000*, Cartagena-Colombia, Aug. 2003. IX Jornadas Iberoamericanas de Informática.

[13] D.B. Skillicorn, J.M.D. Hill, and W.F. McColl. Questions and answers about BSP. Technical Report PRG-TR-15-96, Computing Laboratory, Oxford University, 1996. Also in *Journal of Scientific Programming*, V.6 N.3, 1997.