Simulación y Visualización de la Performance de un Administrador BSP

Paula A. Millado, Daniel O. Laguia, Albert O. Sofia
Universidad Nacional de la Patagonia Austral
Río Gallegos, Argentina (9400)
{pmillado;dlaguia;osofia}@unpa.edu.ar

and

Mauricio Marín
Universidad de Magallanes
Punta Arenas, Chile
mauricio.marin@umag.cl

and

Claudio Delrieux
Universidad Nacional del Sur
Bahía Blanca, Argentina
claudio@acm.org

Abstract

This paper describes a graphic tool for visualizing the behavior of a distributed database server. The target is to provide a graphic tool for the database administrator to evaluate the performance of the database and to suggest a tunning in the broker algorithm, in order to get better request times for the user's queries. We consider a server working with a large query traffic. This workload is served using parallel processing over a cluster with an implementation of the parallel computing BSP model. The visual tool we propose here, allows to show the amount of communication and synchronization time between processors needed by the parallel processing of the queries in the cluster and the workload in database. This information is represented with two visual metaphors, which are useful for the database administrator to take decisions. In this context, the query's order execution can have effects very different at the request time for each query.

Keywords: Databases, Parallel Processing on SQL Queries, Parallel and Distributen Computing, BSP, Cientific Visualization

Resumen

En este trabajo se describe una herramienta gráfica de visualización de la operación de un servidor de bases de datos distribuidas. El objetivo es proporcionar al administrador de la base de datos una herramienta que le permita observar el comportamiento y tomar decisiones orientadas a mejorar los tiempos de respuesta a consultas SQL generadas por los usuarios del sistema. Suponemos un servidor operando con una gran intensidad de tráfico de consultas. Dicha carga de trabajo es servida empleando procesamiento paralelo sobre un *cluster* de PCs, por medio de una implementación del modelo BSP de computación paralela. La herramienta permite visualizar aspectos tales como la cantidad de comunicación y sincronización entre procesadores demandada por el procesamiento paralelo de las consultas. Esa información, adecuadamente representada mediante metáforas visuales, es presentada al administrador para la toma de decisiones. Esto, debido al orden en que los diferentes tipos de consultas a la base de datos generadas por los usuarios son ejecutadas, puede tener efectos muy distintos en el tiempo de respuesta de cada consulta.

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

[1]Este trabajo fue parcialmente financiado por la red CYTED-Ritos2 y por la SECyT-UNS

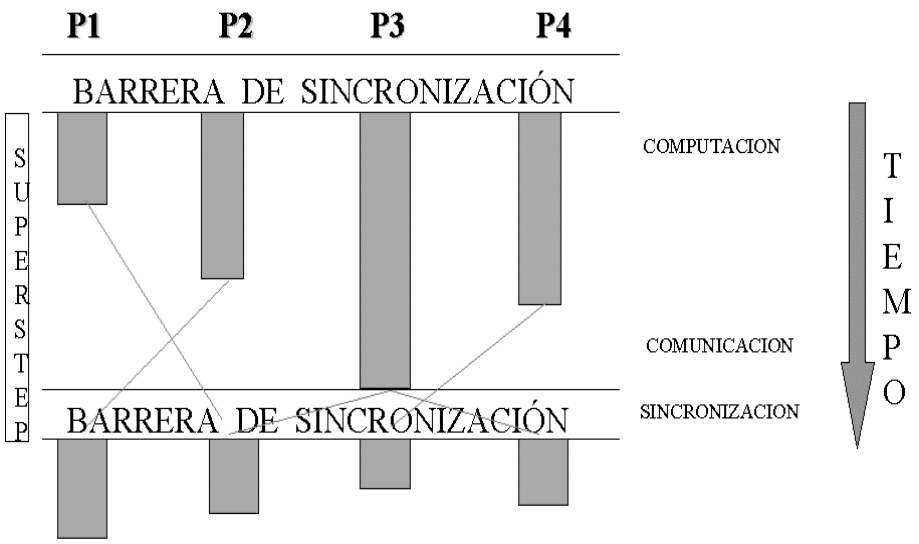

Figura 1: Modelo BSP y supersteps.

1 Modelo de computación paralela BSP

Varios productos comerciales fueron desarrollados para máquinas multiprocesadores de memoria compartida y distribuida [2, 4, 11], y más recientemente para *clusters* de computadores [6, 19, 20]. 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 basada en un modelo relativamente nuevo de computación paralela llamado BSP [21, 25, 28], 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.

En BSP [25] 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*. Durante un superstep, 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. Tal como lo indica la Fig. 1, un superstep está formado por una fase en que los procesadores realizan computaciones sobre datos almacenados en sus memorias locales, tiempo en el cual pueden depositar mensajes a ser enviados a otros procesadores, y al final del superstep se envían todos los mensajes y los procesadores se sincronizan con el objetivo de asegurar que todos han recibido los mensajes, y así comenzar el siguiente superstep.

2 Modelo de Predicción de Costos de BSP

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 de cada superstep está dado por:

1. la suma del costo originado por las computaciones sobre datos locales;
2. el costo de las comunicaciones de mensajes entre procesadores;
3. el costo asociado a la sincronización de los procesadores.

Según lo explicado mas arriba, podemos componer el tiempo de cada superstep como la suma de tres términos: computación + comunicación + sincronización. En otras palabras, el tiempo de ejecución de un superstep s es la suma del máximo tiempo de procesamiento local, el tiempo de envío de los mensajes y el tiempo de sincronización [21, 16]:

$$\text{tiempo}(s) = \text{máx}\{w_i(s)\} + \text{máx}\{h_i(s)\} * g + l, \quad (1)$$

donde $w_i(s)$ es el tiempo de procesamiento local en un procesador i durante el superstep s y $h_i(s) = \text{máx}\{h_{i+}(s), h_{i-}(s)\}$, donde $h_{i+}(s)$ representa el numero de palabras transmitidas y $h_{i-}(s)$ representa el numero de palabras recibidas por el procesador i durante el superstep s. El tiempo de ejecución de un programa BSP está dado por la sumatoria de los tiempos de ejecución de cada superstep ($\sum_s \text{tiempo}(s)$). Por lo tanto el tiempo total de procesamiento puede representarse a través de la fórmula:

$$W + H * g + S * l, \tag{2}$$

donde

$$W = \sum_s \text{máx}\{w_i(s)\}$$

$$H = \sum_s \text{máx}\{h_i(s)\}$$

En general W, H y S son funciones de p (cantidad de procesadores) y n (tamaño del dato). Por lo tanto, el diseño de algotirmos BSP eficientes debe estar guiado por los siguientes criterios:

1. balance de computación en cada procesador, puesto que W es un máximo en computación;
2. balance en comunicación, puesto que $h_{i+}(s)$ y $h_{i-}(s)$ es un máximo sobre los datos enviados o recibidos; y
3. minimización del número total de supersteps requerido para completar el programa.

Los valores de l y g se obtienen en cada implementación de forma empírica, donde l representa el costo de la sincronización y g el costo de transmitir un word desde un procesador a otro en una situación de tráfico continuo en la red de comunicaciones.

3 Configuración del Servidor de Consultas

Actualmente es usual encontrar sitios Web que para su funcionamiento proporcionan acceso a bases de datos relacionales y de texto en la modalidad de cliente-servidor. En tales casos, es deseable que en el lado servidor, el administrador de la base de datos sea capaz de procesar lo más rápido posible las consultas producidas por un número grande de clientes generando peticiones de *Common Gateway Interface* (CGI) a través de lenguajes de *script* tales como PHP o Perl. En particular, un esquema típico puede ser una configuración donde exista una máquina *front-end* que reciba requerimientos desde clientes que ejecutan scripts, la cual a su vez envía instrucciones al servidor de bases de datos y espera por una respuesta del mismo. Estas máquinas *front-end* son las denominadas generadores de consultas. El servidor de base de datos usualmente se comunica con los generadores de consultas por medio de un lenguaje de *querying* como el SQL. Este servidor, además, puede estar alojado en otra máquina y en realidad puede atender simultáneamente los requerimientos provenientes desde dos o más máquinas generadoras de consultas.

Para tal propósito, se ha propuesto anteriormente [10] un servidor de procesamiento de consultas paralelo basado en el modelo *BSP pub* [26], que utiliza una configuración de base de datos distribuida para acelerar las consultas, realizando el procesamiento de la cola de consultas en forma secuencial. Dentro de la gran cantidad de consultas generadas por los generadores de consultas existen diferentes tipologías, las cuales poseen características distintivas de acuerdo a los recursos solicitados, debido fundamentalmente al tiempo de procesamiento requerido y el acceso a la base de datos. Una consulta de alta complejidad podría retrasar sustancialmente a las consultas simples en el esquema de una cola de procesamiento secuencial. El propósito del presente trabajo es presentar una solución gráfica para la representación del desempeño de un servidor paralelo de procesamiento de consultas, que permita asimismo la administración de la cola de consultas a través de la asignación de prioridades y la manipulación del orden de las consultas de la cola. El diseño e implementación de servidores paralelos de sistemas de bases de datos relacionales es un tema que ha sido investigado ampliamente durante la última década [5, 3, 9, 12, 17, 18, 14, 1].

Otro aspecto importante es que nuestra solución se construye a partir de tecnología existente como lo es un servidor secuencial de bases de datos en cada máquina del cluster [27] y el uso de una biblioteca de comunicaciones BSP [26] para gestionar la comunicación y sincronización de las máquinas. Es decir, el costo de implementación y puesta en operación es muy bajo. Proponemos un esquema fácil de implementar y muy eficiente respecto de la alternativa secuencial. El uso de BSP en esta clase de aplicaciones aun no ha sido investigado en profundidad. Trabajos preliminares, principalmente teóricos, en este tema pueden ser encontrados en [13, 23, 22, 24]. En esta línea, este artículo proporciona un enfoque y evaluación práctica de forma visual para el modelo de computación BSP, para lo cual se ha desarrollado una herramienta que permite evaluar el desempeño y administrar la cola de consultas.

La optimización de 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 de la administración del tipo de consultas, generalmente determinada por la programación de un *broker* o agente de gestión. Dentro de la gran cantidad de consultas generadas por los generadores de consultas existen diferentes tipologías, las cuales poseen características distintivas de acuerdo a los recursos solicitados, debido fundamentalmente al tiempo de procesamiento requerido y el acceso a la base de datos. Una consulta de alta complejidad podría retrasar sustancialmente las consultas más simples en el esquema de una cola de procesamiento secuencial. Lamentablemente no hay manera de poder predecir y optimizar estáticamente la administración de las consultas, por lo que una asignatura pendiente es la determinación dinámica de los parámetros que sintonice adecuadamente el funcionamiento optimizado del sistema en una situación particular.

Por dicha razón, en este trabajo exploramos la implementación y uso de una herramienta de visualización para la representación del desempeño del servidor paralelo de procesamiento de consultas, que permita asimismo la administración de la cola de consultas a través de la asignación de prioridades y la manipulación del orden de las consultas dentro de la misma. Para ello, se ha desarrollado un servidor de bases de datos paralelo utilizando el modelo de computación BSP, donde la base de datos se ha distribuido uniformemente. A ese servidor arriban consultas que se discriminan en dos tipologías básicas: consultas simples (correspondientes a *select* simples) y consultas complejas (correspondientes al *join* de dos o más tablas). Una descripción más detallada del diseño del servidor puede ser encontrada en [10, 15].

4 Configuración del Servidor Paralelo de Base de datos

Como ya mencionáramos, nuestra metodología de computación paralela opera sobre un cluster de PCs. El cluster es más bien un accidente que una necesidad puesto que BSP también esta disponible para sistemas multiprocesadores de memoria compartida y distribuida, y los programas pueden ejecutarse sin cambios en cualquiera de estas tres plataformas. En BSP, un computador paralelo es visto como un conjunto de procesadores con memoria local e interconectados a través de una red de comunicación de topología transparente al usuario. La realización práctica de este modelo es a través de una biblioteca de comunicaciones llamada *BSPlib*. Las primitivas de comunicación se invocan desde programas en C y C++, con funciones tales como `bsp_send()` y `bsp_sync()` para comunicar datos y sincronizar las máquinas respectivamente. El modo de programación es SPMD *(simple program multiple data)*. Es decir, un programa BSP es iniciado por el usuario desde una máquina y éste se duplica automáticamente en las máquinas restantes que conforman el cluster. Cada copia ejecuta los mismos superstep pero cada uno opera sobre sus propios datos locales. Adicionalmente, las copias pueden ejecutar caminos alternativos dentro del código de un superstep, pero todos deben alcanzar el mismo punto de sincronización (fin del superstep indicado con `bsp_sync()`) antes de continuar con el siguiente superstep. En nuestro caso, los datos locales son los trozos de la base de datos que se encuentra uniformemente distribuida en los discos de las máquinas que conforman el cluster. Las consultas SQL y las tablas resultado son transmitidas de una máquina a otra mediante arreglos de caracteres enviados utilizando `bsp_send()` y recibidos con `bsp_get()` [21].

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++. Por ejemplo, cada máquina del cluster puede ejecutar comandos SQL sobre su base de datos local mediante instrucciones tales como la siguiente que definimos como *consulta simple*:

```
Connection C("database");
Query Q = C.query();
Q << "select * from PRODUCTOS where disciplina=mensaje.preg";
Result R = Q.store();
cout << "Número total de tuplas recuperadas = " << R.rows() << endl;
```

El siguiente tipo de consulta se define como *Consulta Compleja*:

```
Connection C("database");
Query Q = C.query();
Q << "select sum(cantidad) from PRODUCTOS, VENTAS where PRODUCTOS.codigo= VENTAS.codigo AND
disciplina=mensaje.preg";
Result R = Q.store();
```

```
cout << "Número total de tuplas recuperadas = " << R.rows() << endl;
```

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. Esto se hace mediante una función como *código mod P*, donde *código* es la llave de la tabla. Así, si existen N tuplas para una tabla global dada, estas se encuentran uniformemente distribuidas en las sub-tablas almacenadas en las P máquinas, cada una de ellas manteniendo aproximadamente $\frac{N}{P}$ tuplas. Las tablas no contienen tuplas duplicadas que estén ubicadas en la misma máquina o en otras. En nuestro caso, el generador de consultas va enviando instrucciones SQL de manera circular a cada máquina del cluster BSP. Cuando una máquina i recibe una instrucción SQL, esta transmite a todas las otras una copia de ella para que todas ejecuten la instrucción en sus respectivas bases de datos locales. Luego todas las máquinas transmiten a la máquina i sus resultados parciales de manera que la máquina i pueda construir el resultado final (tabla resultado) y enviárselo como respuesta al generador de consultas. Una implementación BSP de esta estrategia está descripta en el siguiente pseudo-código:

```
// Algoritmo básico ejecutado por cada máquina del cluster.
// Procesamiento paralelo de instrucciones select-from-where.
Inicializa cola de consultas SQL.
while( true )
Comenzar

 Recibe nuevos mensajes desde el generador de consultas SQL, y los almacena en la cola de consult

 Retira un mensaje desde la cola y lo envía a cada máquina del cluster (incluida esta misma,
     cada mensaje contiene el tipo de consulta y una identificación de la máquina que lo envía).

 BSP_SYNC(); (Fin del superstep, sincroniza las máquinas).

 Recibe mensajes desde las máquinas conformando el cluster.

 Por cada mensaje recibido, ejecuta la consulta SQL en el trozo de base de datos almacenados
     en esta máquina.

 Envía la sub-tabla resultante de la consulta a la máquina que originó el mensaje SQL.

 BSP_SYNC(); (Fin del superstep, sincroniza las máquinas).

 Recibe las sub-tablas y las integra para formar la tabla resultado asociada a la consulta
     iniciada por esta máquina.

 Envía la tabla resultado al generador de consultas.

Fin
```

Una característica de las aplicaciones de bases de datos en la Web, es que las tablas resultado que se envían a los clientes en formato HTML son generalmente de tamaño reducido. Esto permite que dichas tablas puedan ser almacenadas en arreglos de caracteres (buffers) en memoria principal. Es decir, la comunicación entre máquinas del cluster, y entre máquinas y generador de consultas, puede ser soportada por buffers de memoria principal sin necesidad de recurrir a técnicas de manejo de almacenamiento secundario. No obstante, en caso de ser necesario transmitir tablas de gran tamaño se pueden destinar superstep adicionales para transmitirlas por partes; cada una de un tamaño igual al máximo espacio disponible en los buffers de comunicación de BSPlib. El algoritmo mostrado en el pseudo-código anterior permite la ejecución concurrente de operaciones de sólo lectura en la base de datos. No es necesario realizar ningún tipo de sincronización de accesos a los datos puesto que estos no sufren modificaciones.

Una observación importante respecto de nuestra metodología de procesamiento paralelo es que los superstep proporcionan un mecanismo global de sincronización de operaciones en el tiempo. Cada ciclo de ejecución de operaciones comienza con el retiro, en cada procesador, de una instrucción desde la cola de consultas SQL, lo cual es seguido por una comunicación global. El generador de consultas envía instrucciones SQL consecutivamente. El punto aquí es cubrir el mayor espectro posible sin llegar a tener que construir un administrador de bases de datos paralelo o modificar uno secuencial. Enfatizamos que la solución aquí propuesta utiliza tecnología existente de bajo costo, un DBMS secuencial como MySQL [27], un grupo de PCs conectados mediante un switch, C++ y una biblioteca de comunicaciones para el modelo BSP de computación paralela llamada BSPpub [26].

Por supuesto, la eficiencia de este esquema depende de una adecuada distribución de los datos en las máquinas que conforman el cluster. Para esto el modelo BSP proporciona una forma de cuantificar el costo en computación, comunicación y sincronización de programas BSP. En [16] proponemos una extensión a dicho modelo con el objeto de ayudar al diseñador de la base de datos en la decisión por una determinada distribución tanto de tablas como de tuplas de dichas tablas en los discos de las máquinas del cluster.

Anteriores experimentos con esta configuración han demostrado una mejora altamente significativa en la performance de las consultas. En [10] realizamos la evaluación del desempeño de esta configuración. Sin embargo, debería ser posible aún mejorar el rendimiento ajustando parámetros y características de la cola de consultas SQL, de manera de tomar en cuenta el tiempo de espera de los clientes que han generado las consultas para maximizar su satisfacción al realizar un acceso a la aplicación. En la siguiente sección presentamos una descripción de la herramienta desarrollada para evaluar gráficamente el desempeño del servidor de consultas y administrar ciertas características de la cola de consultas SQL. En todos los casos se utiliza un benchmark basado en dos tipos de instrucciones select, puesto que como se explicó anteriormente en esta sección, con este tipo de instrucciones es más difícil obtener buen desempeño en el caso paralelo puesto que demandan mayor comunicación (envío de tablas de resultados parciales) que instrucciones como insert y update, las cuales no generan tablas de resultados.

5 Optimización Dinámica

El propósito de la visualización del desempeño del sistema es que el administrador pueda diagnosticar visualmente un desbalance en la carga que está recibiendo cada procesador. En ese contexto, el objetivo de esta herramienta es, además de observar el comportamiento de la base de datos paralela en tiempo real, permitir la administración de la misma, de manera de poder mejorar la performance del sistema. Esta optimización dinámica se logra operando sobre la cola de consultas, trabajando básicamente con dos colas, correspondientes una a las consultas simples y otra a las complejas. En nuestro caso hemos permitido al operador la posibilidad de asignar prioridades en la ejecución de las consultas, mediante la definición de un esquema de proporcionalidad en la relación consultas simples/consultas complejas.

Esta proporción indica una prioridad debido a que, en caso de existir en la cola de consultas operaciones simples y complejas, los clientes que han solicitado una consulta simple se verían retrasados en forma muy considerable si existiera una sucesión de varias consultas complejas. Aunque estas últimas se hayan producido con anterioridad, sería un esquema desventajoso para los clientes con consultas simples si se realizara la cola solo en forma secuencial. En el otro extremo, posponer las consultas complejas hasta la finalización de todas las consultas simples podría implicar una demora excesiva para las primeras. Si bien ésta es una solución sencilla, podrían estudiarse otras alternativas para optimizar la performance de la base de datos. Sin embargo es dificultoso manipular en tiempo real los parámetros del modelo de costos BSP, ya que implica la modificación del código fuente de los programas utilizados en el servidor.

6 Metáforas Visuales

En trabajos anteriores [7], desarrollamos el primer prototipo de visualización del desempeño que permitía una correcta administración del servidor paralelo. En esta primera aproximación se representaba la performance global de los procesadores en los parámetros funcionales más importantes para el servidor (Tiempo de Procesamiento TP, Tiempo de Sincronización TS, Tiempo de Acceso a la Base de Datos TB y Tiempo de Comunicación TC). Sin embargo, esta percepción global de los sucesos producidos en la cola de consultas es inadecuada, debido principalmente a la falta de granularidad en la visualización de los supersteps. Por dicha razón, una de las mejoras necesarias consiste en poder observar y comparar esta información con la correspondiente a otros supersteps, obteniendo por lo tanto un historial de la performance del sistema desde una base temporal determinada. Para ello, en el presente trabajo desarrollamos dos nuevas maneras de visualizar la performance del sistema.

En la primera metáfora visual es una evolución natural de los trabajos anteriores (ver Fig. 2). Aquí se optó por representar el comportamiento del sistema graficando la performance de cada procesador en cada superstep. De esa manera, el comportamiento de cada procesador está representado con una torre, en la cual cada uno de sus pisos es un superstep. El color de dicho piso representa la performance del procesador dado. Esto se logró mediante la graficación de los valores de los parámetros funcionales en una paleta bidimensional que representa, en el modelo de costos del modelo BSP, la relación entre el tiempo de comunicación y sincronización respecto del tiempo de procesamiento (ver Fig. 3). Una carga balanceada se caracteriza por tener $TP + TB$ comparable a $TS + TC$, no importando si estos valores son bajos o altos. La paleta, por lo tanto, permite además visualizar esta correlación. Un procesador que en un superstep insume mayor $TP + TB$ se representa con un color naranja, mientras que si $TS + TC$ es mayor, el color tiende al

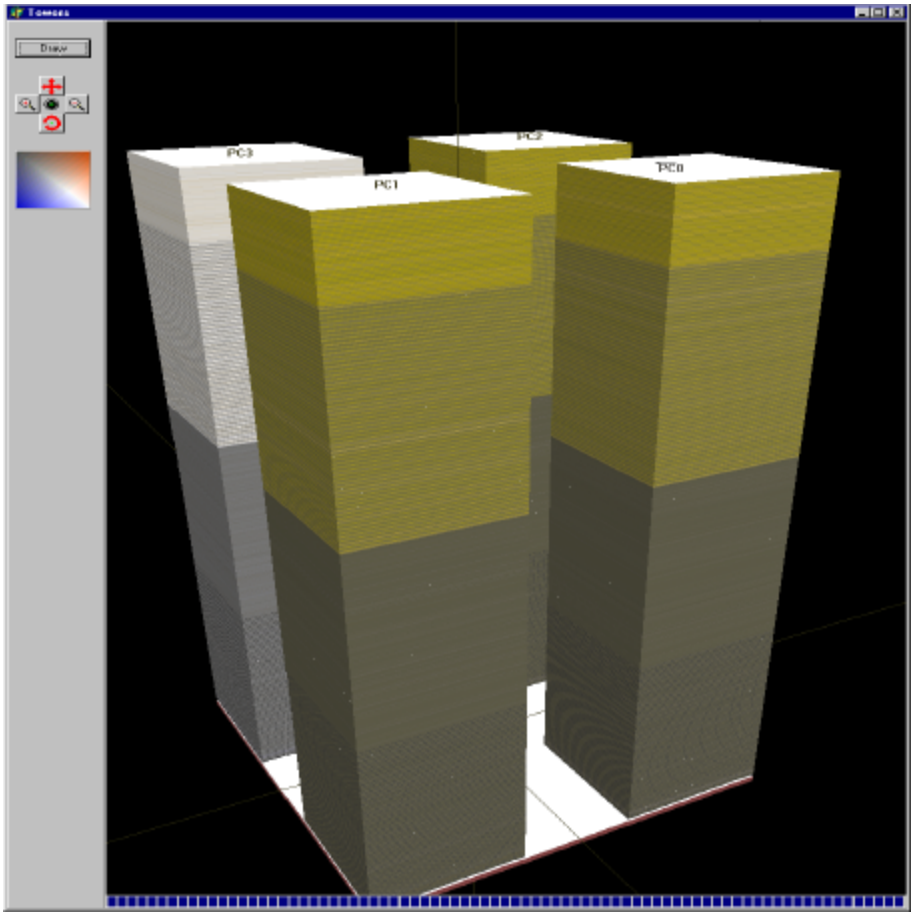

Figura 2: Sucesión de supersteps representada como *pisos* por orden de llegada. La performance de cada superstep se representa por medio de una paleta (ver Fig. 3).

azul. Si ambos tiempos están equilibrados, el color es neutro. Si ambos tiempos están equilibrados y son bajos, el color tiende al blanco, y si ambos son altos, tiende al negro. En la Fig. 2 se puede visualizar el cambio en la performance de los procesadores a medida que la proporción de consultas se va modificando.

Esta paleta bivariada se consigue asignando luminancia y saturación positivas en una de las variables a representar, y luminancia positiva y saturación "negativa" en la otra variable, interpretando la saturación negativa como saturación en la crominancia complementaria. De esa manera, valores altos de las variables sinergizan en producir una luminancia alta, pero compiten en crominancia, produciendo un color neutro. De hecho, la paleta muestra adecuadamente la zona de valores donde existe correlación entre ambas variables.

En base a la ecuación 2 puede obtenerse una relación entre el tiempo de ejecución de un superstep y el tiempo comparable de un procedimiento secuencial, dado por la siguiente fórmula:

$$\frac{T_{\text{secuencial}}}{T_{\text{paralelo}}} = \frac{w_1 + w_2 + w_3 + w_4}{W_{\text{máx}} + h * G + M_s * L} = P, \tag{3}$$

lo cual representa una aproximación al *speedup* del modelo paralelo al trabajar con P procesadores. Haciendo referencia al procedimiento anterior:

$$v_1 = \frac{W_{\text{máx}}}{TP_{\text{máx}} + TB_{\text{máx}}}, \tag{4}$$

donde $W_{\text{máx}} = \text{máx}[TP + TB]$.

$$v_2 = \frac{TC + TS}{TC_{\text{máx}} + TS_{\text{máx}}} \tag{5}$$

$$v_3 = \frac{T_{\text{secuencial}}}{(W_{\text{máx}} + TC + TS) * P} \tag{6}$$

Aqui, v_1 representa la relación entre el tiempo de procesamiento mayor de un superstep y los máximos obtenidos de esos valores desde la base temporal determinada. v_2 representa la relación entre el tiempo

Figura 3: Paleta de Colores.

de Comunicación y Sincronización de un superstep y los máximos obtenidos de esos valores desde la base temporal determinada. v_3 representa la relación entre el tiempo secuencial y el tiempo paralelo de un superstep.

Con esta base, se realizó una serie de experimentos para evaluar el comportamiento de la herramienta, comenzando con una proporción alta de consultas simples. A continuación, luego de un cierto número de consultas se detecta el aumento de la proporción de consultas complejas con la consiguiente pérdida de performance de la Base de Datos. Cuando el administrador detecta dicha caída interviene ampliando la proporción de consultas simples, con lo que se restablece la performance anterior (representado por los colores neutros, gris en este caso, ver Fig. 2). Por otra parte, podemos observar en dicha figura la diferencia de performance de cada uno de los procesadores, por ejemplo el procesador 3 posee mayores capacidades de procesamiento, lo que se ve reflejado con colores más neutros (su TB es menor al de los otros tres procesadores).

La segunda metáfora visual explorada (ver Fig. 4) permite un análisis más minucioso de la historia del procesamiento de una serie de consultas. Ésta consiste en representar la performance de cada procesador en "cintas" verticales correspondientes a las variables del modelo de costos. A su vez cada cinta se encuentra dividida en carriles, representando a cada uno de los procesadores involucrados (P0, P1, P2 y P3 en nuestro caso). Cada bloque representa el valor de una variable en particular para un superstep o conjunto de supersteps dado. También en esta figura es posible observar el cambio en la performance de los procesadores a medida que va modificándose la proporción de consultas simples y complejas.

Para facilitar la visualización pueden agruparse varios de ellos en un solo bloque, disminuyendo la granularidad pero ofreciendo una imagen más integral sobre la performance de la base de datos distribuida (ver Fig.5). El tiempo está representado por la longitud de las cintas, donde el bloque inferior representa el conjunto de superstep más reciente. En la figura se puede observar cómo una visualización de menor granularidad permite observar detalles más ricos de la performance de los procesadores en instantes puntuales de la ejecución.

Por otra parte puede configurarse la paleta de colores para obtener representaciones más intuitivas. Por ejemplo, seleccionando una gama de crominancia que abarque desde el verde al rojo, donde rojo representa los valores de variables más elevados y verde representa los valores más bajos. Por último, puede configurarse la saturación, y luminancia, con las cuales podemos dar mayor énfasis a determinadas zonas de la paleta, donde se desea que el operador preste inmediatamente atención (ver Fig. 4) [29, 8].

Con los mismos datos que se utilizaron en la Fig. 4, en la Fig.5 pueden observarse claramente las diferencias entre los procesadores P0, P1 y P2 con respecto a P3, relacionados fundamentalmente con la performance de cada uno de los procesadores y los estados del servidor de procesamiento de consultas a lo largo de la ejecución del experimento como resultado del cambio de parámetros en el mismo. En los bloques más recientes se observa una baja performance de los procesadores P0, P1 y P2 en TB, como consecuencia de la definición de una alta proporción de consultas complejas en la cola de consultas. Consecuentemente P3, un procesador más potente, posee un TS mucho mayor, debido a que ha terminado sus tareas de procesamiento con anterioridad a los restantes procesadores. En este experimento se aumentó progresivamente la proporción de consultas complejas sobre las consultas simples.

7 Conclusiones

Un aspecto clave en la solución propuesta para el servidor es que éste esta construido sobre el modelo de computación paralela BSP. Es sabido que este modelo soporta una metodología estructurada de diseño de

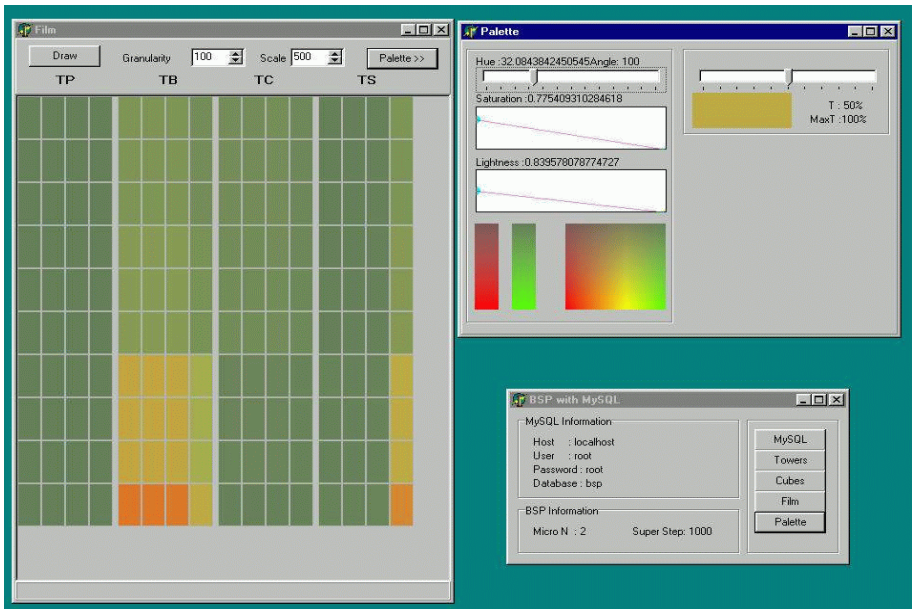

Figura 4: Sucesión de supersteps representada como línea de tiempo vertical.

Figura 5: Diferentes granularidades en la visualización.

software que es simple de utilizar. Igualmente importante es el hecho de que la estructura del modelo permite cuantificar el costo de ejecución del software localizándose no solo en la componente de computación sino que también en las componentes de comunicación y sincronización. Existe una manera explicita de realizar esta cuantificación, la cual consiste en contabilizar la cantidad de computación y comunicación realizada en cada superstep, y luego sumar sobre todos los supersteps ejecutados.

La principal contribución de este trabajo está precisamente en la utilización del modelo BSP y su método de cuantificación de tiempo de ejecución para formular una herramienta gráfica que permite visualizar medidas de desempeño y en base a dicha visualización tomar decisiones respecto de la operación del servidor. En particular, en este trabajo nos hemos concentrado en el orden en que es conveniente procesar distintos tipos de consultas SQL que están arribando al servidor a una cierta tasa de llegadas. El objetivo es proporcionar al administrador de la base de datos una forma simple de visualizar lo que esta ocurriendo con el tráfico de consultas en un periodo dado de la operación del sistema con el fin de realizar optimizaciones.

Referencias

[1] ACM Computing Surveys. *Distributed query processing*, Dec. 1984.

[2] R. Bamford. Architecture of oracle parallel server. *VLDB'98*, pages 669–670, 1998.

[3] N. S. Barghouti and G. E. Kaiser. Concurrency control in advanced database applications. *ACM Computing Surveys*, pages 269–317, 1991.

[4] C. Baru, G. Fecteau, and A. Goya. Db2 parallel edition. *IBM Systems Journal*, pages 292–322, 1995.

[5] D. Bitton, H. Boral, D. J. DeWitt, and W. K. Wilkinson. Parallel algorithms for the execution of relational database operations. *ACM Transactions on Database Systems*, pages 324–353, Sep. 1983.

[6] G. Bozas, M. Jaedicke, and A. Listl. On transforming a sequential sql-dbms into a parallel one: First results and experiences of the midas project. *EuroPar'96*, pages 881–886, Aug. 1996.

[7] CACIC 2003. *Representación visual para la administración del procesamiento paralelo de consultas SQL*, La Plata - Argentina, Octubre 2003.

[8] C. Delrieux. Fundamentos e implementación de sistemas de computación gráfica. Departamento de Ingenieria Eléctrica, Universidad Nacional del Sur, 1999.

[9] D. DeWitt and J. Gray. Parallel database systems: The future of high performance database systems. *Comunications of the ACM*, pages 85–98, June. 1992.

[10] En Jornadas Chilenas de Computación 2001. *Procesamiento paralelo de consultas SQL generadas desde la Web*, Nov. 2001.

[11] B. Gerber. Informix on line xps. *ACM SIGMOD'95*, 24 of SIGMOD Records:463, May. 1995.

[12] G. Graefe. Query evaluation techniques for large databases. *ACM Computing Surveys*, page 73:170, June 1993.

[13] J.M.D. Hill, S. A. Jarvis, C. Siniolakis, and V. P. Vasilev. Portable and architecture independent parallel performance tuning using a call-graph profiling tool: A case study in optimising sql. Technical Report PRG-TR-17-97, Computing Laboratory, Oxford University, 1997.

[14] In 1997 International Conference on Parallel and Distributed Computing Systems. *Architectures for parallel query processing on networks of workstation.*, Oct. 1997.

[15] IX Jornadas Iberoamericanas de Informática. *Visualización Gráfica de Consultas SQL en Paralelo*, Cartagena-Colombia, Agosto 2003.

[16] M. Marín, J. Canumán, and D. Laguía. Un modelo de predicción de desempeño para bases de datos relacionales paralelas sobre bsp. *VI Congreso Argentino de Ciencia de la Computación*, Oct 2000.

[17] P. Mishra and M. Eich. Join processing in relational databases. *ACM Computing Surveys*, March 1992.

[18] C. Mohan, H. Pirahesh, W G. Tang, and Y. Wang. Parallelism in relational database management systems. *IBM Systems Journal*, pages 349–371, 1994.

[19] Oracle. *Oracle parallel server: Solutions for mission critical computing. Technical Report Oracle Corp.*, Feb. 1999.

[20] SC'97. *Parallel database processing on a 100 node pc cluster: Cases for decision support query processing and data mining*, 1997.

[21] 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.

[22] K. R. Sujithan. Scalable high-performance database servers. In *2nd IEE/BCS International Seminar on Client/Server Computing*, May 1997. IEE Press.

[23] K.R. Sujithan. Towards a scalable parallel object database — the bulk-synchronous parallel approach. Technical Report PRG-TR-17-96, Computing Laboratory, Oxford University, 1996.

[24] K.R. Sujithan and J. M. D. Hill. Collection types for database programming in the bsp model. In *5th EuroMicro Workshop on Parallel and Distributed Processing (PDP'97)*, Jan. 1997. (IEEE CS Press).

[25] URL. BSP and Worldwide Standard, http://www.bsp-worldwide.org/.

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

[27] URL. MySQL Web Page, http://www.mysql.com/.

[28] L.G. Valiant. A bridging model for parallel computation. *Comm. ACM*, 33:103–111, Aug. 1990.

[29] Alan Watt. *3D Computer Graphics*. Addison - Wesley, 2da edition, 1993.