Administrador Visual de Entornos BSP

Paula Millado [1], Daniel Laguía [1], Osiris Sofía [1], Mauricio Marín [1,2] y Claudio Delrieux [1,3]

[1]Universidad Nacional de la Patagonia Austral, Argentina
[2]Universidad de Magallanes, Chile
[3]Universidad Nacional del Sur, Argentina

1. Modelo BSP y Configuración del Servidor Paralelo de Base de datos

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.

El diseño e implementación de servidores paralelos de bases de datos relacionales es un tema que ha sido investigado ampliamente durante la última década [5, 3, 7, 16, 9, 13, 12, 24]. Han sido desarrollados varios productos comerciales para máquinas multiprocesadores de memoria compartida y distribuida [4, 2, 8], y más recientemente para clusters de computadores [6, 14, 22]. Todos estos desarrollos están basados en modelos tradicionales de computación paralela como por ejemplo pasaje de mensajes o memoria compartida. En este trabajo se presenta una solución basada en un modelo relativamente nuevo de computación paralela llamado BSP [18, 17, 23], 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. Un punto importante del modelo BSP es que la sincronización entre procesadores se realiza por medio de *supersteps*, es decir, el trabajo de las consultas se distribuye, se procesa en forma paralela en cada procesador, y al finalizar el *superstep* los resultados son propagados.

Un aspecto notable de esta solución es que la misma se construye a partir de tecnología existente como lo es un servidor secuencial de bases de datos en cada máquina del cluster [15] y el uso de una biblioteca de comunicaciones BSP [1] para gestionar la comunicación y sincronización de las máquinas. Es decir, el costo de implementación y puesta en operación es muy bajo. El uso de BSP en esta clase de aplicaciones aun no ha sido investigado en profundidad, aunque algunos trabajos preliminares, principalmente teóricos, en este tema pueden ser encontrados en [10, 19, 20, 21].

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.

El propósito del presente artículo es presentar 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 la realización de este trabajo 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 mas tablas). Una descripción mas detallada del diseño del servidor puede ser encontrada en [11,25, 26, 27].

2. 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 el 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 [1], ya que implica la modificación del código fuente de los programas utilizados en el servidor.

3. Metáforas Visuales

En trabajos anteriores [26, 27] 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. 1). 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 representación 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. Una carga balanceada se caracteriza por tener TP comparable a TS + TC, no importando si estos valores son bajos o altos. La paleta, por lo tanto, permite además visualizar

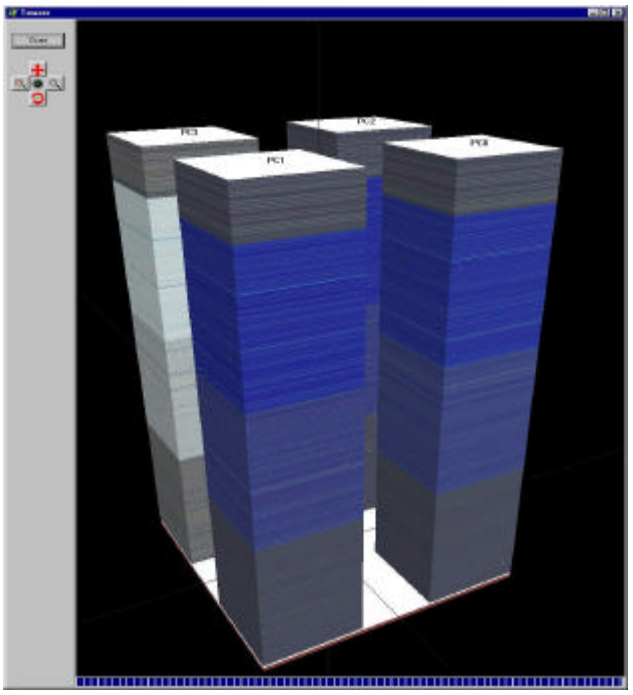

Figura 1: La primera metáfora visual.

esta correlación. Un procesador que en un superstep insume mayor TP se representa con un color naranja, mientras que si TS + TC es mayor, el color tiende al 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.

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. 1). 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 TC es menor al de los otros tres procesadores).

La segunda metáfora visual explorada (ver Fig. 2) permite un análisis más minucioso de la historia del procesamiento de una serie de consultas. Esta 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. 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. 3). El tiempo está representado por la longitud de las cintas, donde el bloque inferior representa el conjunto de SuperStep más reciente.

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 verdes 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. 2) [28, 29].

Con los mismos datos que se utilizaron en la Fig. 1, en la Fig. 2 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

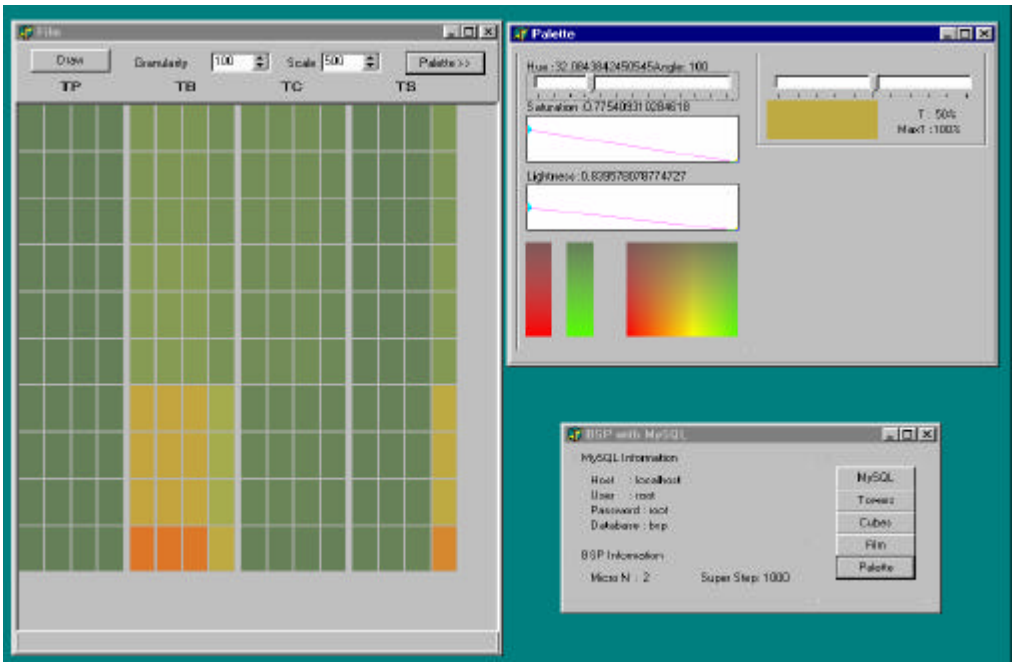

Figura 2: La segunda metáfora visual.

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.

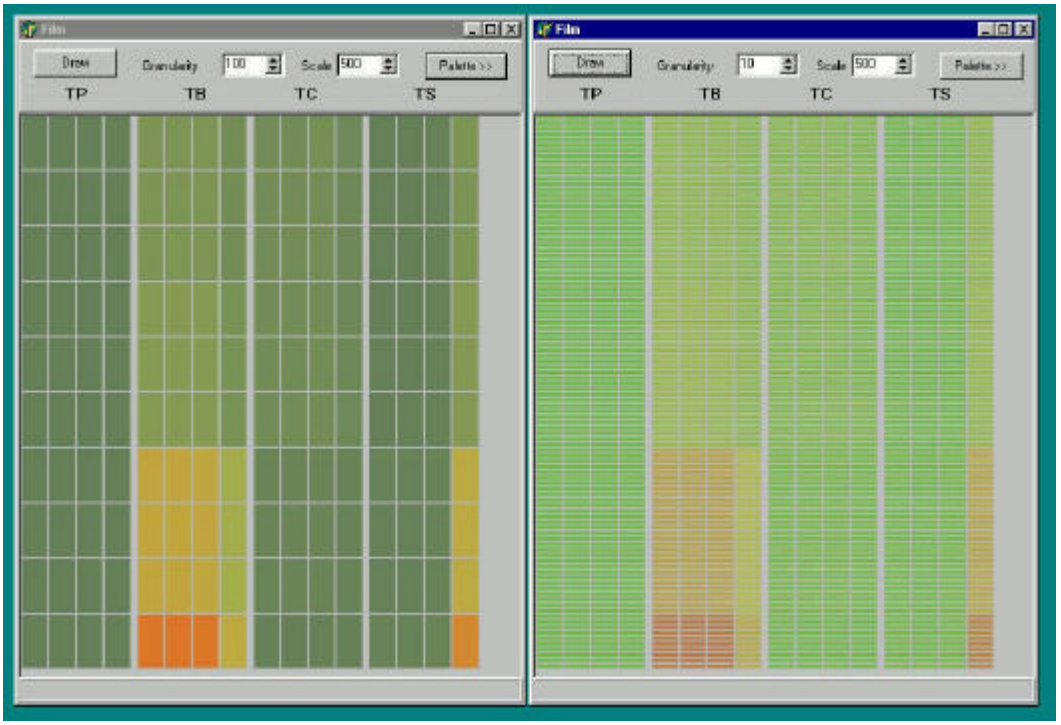

Figura 3: Diferentes granularidades en la visualización.

4. 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 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] PUB BSP Library at Paderborn University. ~ http://www.uni-paderborn.de/bsp.
[2] R. Bamford et al. Architecture of oracle parallel server. In VLDB'98, pages 669:670, Aug. 1998.
[3] N. S. Barghouti and G. E. Kaiser. Concurrency control in advanced database applications. ACM Computing Surveys, 23(3):269:317, Sept. 1991.
[4] C. Baru, G. Fecteau, and A. Goya et al. Db2 parallel edition. IBM Systems Journal, 34(2):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, 8(3):324:353, Sept. 1983.
[6] G. Bozas, M. Jaedicke, and A. Listl et al. On transforming a sequential sql-dbms into a parallel one: First results and experiences of the midas project. In EuroPar'96, pages 881:886, Aug. 1996.
[7] D. DeWitt and J. Gray. Parallel database systems: The future of high performance database systems. Comunications of the ACM, 35(6):85:98, June 1992.
[8] B. Gerber. Informix on line xps. In ACM SIGMOD'95, May 1995. Vol 24. of SIGMOD Records, p. 463.
[9] G. Graefe. Query evaluation techniques for large databases. ACM Computing Surveys, 25(2):73:170, June 1993.
[10] J.M.D. Hill, S. A. Jarvis, C. Siniolakis, and V. P. Vasilev. Portable and architecture independent parallel performance tuning using a call-graph proling tool: A case study in optimizing sql. Technical Report PRG-TR-17-97, Computing Laboratory, Oxford University, 1997.
[11] M. Marín, J. Canumán, and D. Laguía. Un modelo de predicción de desempeño para bases de datos relacionales paralelas sobre BSP. In VI Congreso Argentino de Ciencia de la Computación, Oct. 2000.
[12] P. Mishra and M. Eich. Join processing in relational databases. ACM Computing Surveys,24(1):63:113, March 1992.
[13] C. Mohan, H. Pirahesh, W G. Tang, and Y. Wang. Parallelism in relational database management systems. IBM Systems Journal, 33(2):349:371, 1994.
[14] Oracle. Oracle parallel server: Solutions for mission critical computing. Technical Report Oracle Corp., Feb. 1999.
[15] MySQL Web Page. http://www.mysql.com/.
[16] S.Dandamudi and J.Jain. Architectures for parallel query processing on networks of workstation. In 1997 International Conference on Parallel and Distributed Computing Systems, Oct.1997.
[17] 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 Scientic Programming, V.6 N.3, 1997.
[18] BSP Worldwide Standard. http://www.bsp-worldwide.org/.
[19] K. R. Sujithan. Towards a scalable parallel object database | thebulk-synchronous parallel approach. Technical Report PRG-TR-17-96, Computing Laboratory, Oxford University, Aug.1996.
[20] K. R. Sujithan. Scalable high-performance database servers. In 2nd IEE/BCS International Seminar on Client/Server Computing, May 1997. IEE Press.
[21] 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.
[22] T. Tamura, M. Oguchi, and M. Kitsuregawa. Parallel database processing on a 100 node pc cluster: Cases for decision support query processing and data mining. In SC'97, 1997.
[23] L.G. Valiant. A bridging model for parallel computation". Comm. ACM, 33:103{111, Aug.1990.
[24] C. T. Yu and C. C. Chang. Distributed query processing. ACM Computing Surveys, 16(4):399, 433, Dec. 1984.
[25] M. Marin, J. Canuman, M. Becerra, D. Laguia, O. Sofia. Procesamiento paralelo de consultas SQL generadas desde la Web. En Jornadas Chilenas de Computación 2001, Punta Arenas, Chile, Nov. 2001.
[26] Paula Millado, Daniel Laguia, Albert Sofia, Mauricio Marín, "Visualización Gráfica de Consultas SQL en Paralelo", IX Jornadas Iberoamericanas de Informática- Cartagena- Colombia - Agosto 2003.
[27] Paula Millado, Daniel Laguia, Albert Sofia, Mauricio Marín, "Representación visual para la administración del procesamiento paralelo de consultas SQL", CACIC 2003- La Plata- Argentina - Octubre 2003.
[28] Alan Watt, "3D Computer Graphics", Segunda Edición, Ed Addison – Wesley, 1993.
[29] Claudio Delrieux, "Fundamentos e Implementación de Sistemas de Computación Gráfica", Departamento de Ingenieria Eléctrica, Universidad Nacional del Sur, 1999.