Procesamiento de Consultas Distribuidas en Base de Datos Espacio-Temporales

Claudio Gutiérrez-Soto

Departamento de Sistemas de Información, Universidad del Bío-Bío, Concepción, Chile, cogutier@ubiobio.cl

Rodrigo Osorio

Departamento de Sistemas de Información, Universidad del Bío-Bío, Concepción, Chile, rosorio@alumnos.ubiobio.cl

Leonardo Ramos

Departamento de Sistemas de Información, Universidad del Bío-Bío, Concepción, Chile, leramos@alumnos.ubiobio.cl

Abstract

Spatio-temporal databases have reached great interest. However, there is a scarce work of these databases in distributed environments. This paper describes two parallel strategies for the MVR-tree spatio-temporal access method on CREW PRAM parallel model. Experiments to compare our strategies with sequential approximation were carried out. Preliminary results show significant savings both in accessed nodes as well as in the execution time in a CREW PRAM parallel environment.

Key words: Spatio-temporal databases, Parallel Computing.

Resumen

Las Bases de Datos Espacio-Temporales han logrado una gran atención. Sin embargo existe un escaso trabajo de dichas base de datos en ambientes distribuidos. En este artículo se describen dos estrategias de paralelización del método de acceso espacio temporal MVR-tree sobre el modelo paralelo CREW PRAM. Se realizaron experimentos que comparan nuestras estrategias con el enfoque secuencial. Los resultados preliminares muestran que se produce un ahorro significativo de nodos accesados, así como de los tiempos de procesamiento en un ambiente paralelo CREW PRAM.

Palabras Claves: Base de Datos Espacio-Temporales, Paralelismo.

1. Introducción

Las Base de Datos Espacio-Temporales han alcanzado un gran auge, esto debido a que poseen dos atributos inherentes en los objetos del mundo real. Dichos atributos son el tiempo y el espacio. Así, un objeto es caracterizado por sus posiciones a lo largo del tiempo [1]. Este tipo de objetos hace necesario tener datos espacio-temporales, los cuales puedan ser manejados por aplicaciones computacionales. Un ejemplo de este tipo de aplicación puede ser una flota de taxis, en donde se necesita saber la posición de ellos en un determinado instante de tiempo. Esto permitiría responder preguntas tales como ¿cuáles son los automóviles que están más cercanos a la plaza de armas? o ¿Cuáles son los autos más cercanos al BB-3455(El cual podría requerir asistencia)[2].

Los tipos de consultas espacio-temporales más estudiadas son las del tipo *time-slice* y *time-interval* [3]. Las consultas del tipo *time-slice* recuperan todos los objetos que se intersectan espacialmente con el rango espacial, dado en la consulta en un instante particular de tiempo. Las consultas del tipo *time-interval* extienden la idea de las consultas del tipo *time-slice* considerando instantes de tiempo consecutivos.

Dentro de las características de las Bases de Datos Espacio-Temporales podemos destacar su gran volumen y como consecuencia de ello el alto costo de los algoritmos para evaluar las consultas. Una forma de mejorar el desempeño de este tipo de bases de datos es por medio de su distribución (Base de Datos Distribuidas) y paralelización de los algoritmos utilizados para manejarla. En [4] se presenta la paralelización de los métodos R-Tree y MVR-tree, donde el objetivo principal es obtener un mecanismo de control de concurrencia para un Data Warehouse. Sin embargo, dicha paralelización se basa fundamentalmente en la capacidad de almacenamiento, por lo que en dicho trabajo se omite la eficiencia de la paralelización de estos métodos. Por otro lado, en [5] se presenta un modelo de costos para consultas del tipo *time-interval* utilizando el índice de acceso MVR-tree bajo el modelo paralelo CREW PRAM. Sin embargo, en dicho trabajo se omiten otras estrategias y resultados empíricos de consultas del tipo *time-slice*.

La principal contribución de este trabajo es la evaluación empírica de dos estrategias simples de paralelización sobre el índice de acceso MVR-tree para consultas del tipo *time-slice* y *time-interval*. Por otro lado, se realiza una evaluación del modelo de costos propuesto por [5], para consultas del tipo *time-slice* considerando los resultados empíricos de las estrategias propuestas.

El resto del artículo está organizado de la siguiente manera, en la sección 2 se entrega una descripción de los métodos de acceso para Base de Datos Espacio-Temporales enfocado principalmente en la estructura del MVR-tree, en la sección 3 se obtiene un modelo de costos del MVR-tree en el enfoque CREW PRAM. En la sección 4 discuten resultados experimentales sobre la estrategia de paralelización. Finalmente, se dan las conclusiones y el trabajo futuro.

2. Métodos de Acceso Espacio-Temporales

Uno de los métodos de acceso multidimensional o espacial más estudiados y, por ende, uno de los más conocidos y que ha sido adoptado por productos de SABD (Sistemas de Administración de Bases de Datos), tales como Oracle y Postgres es el R-tree. Un R-tree es una extensión natural de un B-tree [6] para objetos espaciales (puntos y regiones) [7]. En un R-tree un nodo corresponde a una página o bloque de disco. Los nodos hojas de un R-tree contienen entradas de la forma <MBR,oid> donde oid es el identificador del objeto espacial en la base de datos y MBR (Minimun Bounding Rectangle) es un rectángulo multidimensional que corresponde al mínimo rectángulo que

encierra al objeto espacial. Los nodos internos (nodos no-hojas) contienen entradas de la forma <MBR, ref>, donde ref es la dirección del correspondiente nodo hijo en el R-tree y MBR es el rectángulo mínimo que contiene a todos los rectángulos definidos en las entradas del nodo hijo.

La Figura. 1.1. (a) ilustra cuatro 2D rectángulos R_1, R_2, R_3 y R_4 junto con los nodos MBR de los correspondientes R-tree (nodos con capacidad igual a 2) los cuales son mostrados en la Figura. 1.1. (b) Basados en su proximidad espacial, R_1 y R_2 son agrupados juntos en el nodo N_1 (cuyo padre está en R_5) y R_3, R_4 están en N_2 (cuyo padre está R_6). Dado una *consulta por rango espacial o window query* q_R (el rectángulo gris de la Figura. 1.1. (a), los objetos recuperados para dicha consulta (R_1, R_2 y R_3) son aquellos nodos cuyos MBR se intersectan con q_R en el ejemplo.

Un método de acceso espacio-temporal, permite recuperar objetos cuyo atributo espacial cambia su posición o forma a lo largo del tiempo.

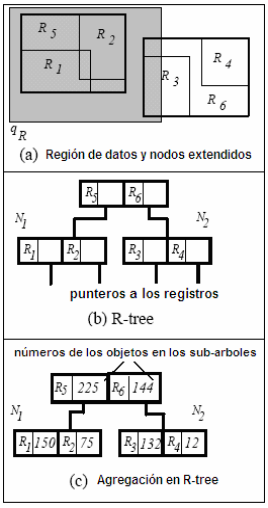

Figura 1.1: Ejemplo de R-tree

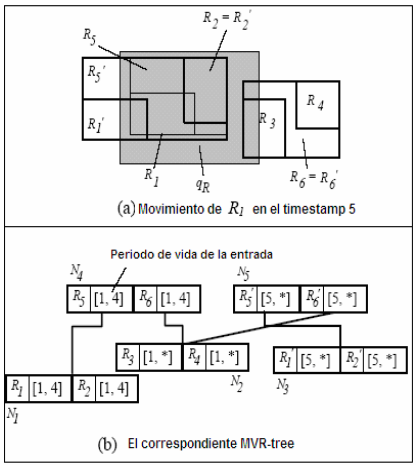

Figura 1.2: MVR-tree.

Figura. 1: Ejemplo de R-tree y MVR-tree

Un método de acceso a Bases de Datos Espacio-Temporal es el Multi-versión R-tree (MVR-tree) [8]. En el MVR-tree una entrada tiene la forma <MBR, t_{st} ,t_{ed}, ref>, donde [t_{st} ,t_{ed}] denota el tiempo de vida, por ejemplo, el intervalo de tiempo durante el cual un objeto se encuentra vivo (t_{ed}="*" implica que la entrada se encuentra aún viva o en la misma posición en el tiempo actual). En cada entrada de un nodo hoja MBR denota los MBR de los objetos, mientras que las entradas en los nodos internos cubren todas las entradas de los hijos vivos que se encuentran cercanas con sus correspondientes periodos de vida. La semántica de r.ref es similar a los R-tree ordinarios.

La Figura. 1.2. (a) muestra un ejemplo donde R_1 se mueve a una nueva posición R_1' en el instante de tiempo 5 (lo que hace que R_5 cambie a R_5'), y la Figura. 1.2. (b) ilustra el correspondiente *MVR-tree.* Los R-trees (lógicos) para el intervalo de tiempo [1,4] cubren entradas en los nodos N_1, N_2, N_4 (observe los periodos de vida de la entrada de los padres), mientras se comienza desde el instante de tiempo 5, los árboles lógicos consisten de los nodos N_5 y N_3, los cuales reemplazan a N_4 y N_1 respectivamente. Note que N_2 es compartido (este es el hijo de los nodos N_4 y N_5) porque ninguno de sus objetos cambió su posición. El algoritmo para evaluar consultas de tipo *time-slice* o *time-interval* del MVR-tree es el mismo que del R-tree normal, excepto que la búsqueda es ejecutada en los posibles árboles lógicos para la consulta.

2.1 Modelo de Costo para el MVR-tree

En esta sección se describe el modelo de costo asociado a la estructura del MVR-tree, el cual nos permite predecir los costos de las consultas espacio-temporales en un ambiente paralelo bajo el modelo CREW PRAM. Este modelo de costo asume que los movimientos de los objetos están uniformemente distribuidos en el tiempo.

2.1.1 Costo de consultas en un MVR-tree

En esta sección se resume el modelo de costo para el método de acceso MVR-tree definido en [3]. Dicho modelo permite estimar el rendimiento de las consultas de un MVR-tree.

Sin pérdida de generalidad, asumimos que el atributo espacial de los objetos están distribuidos en el universo $[0,1]^d$. Donde *d* es la dimensionalidad del espacio (para un B-Tree, d=1). Una consulta del tipo *time-interval q* puede estar representado como $q(q_k, q_t)$, para indicar el rango espacial q_k, y el intervalo temporal q_t.

Sea *N* el número de objetos almacenados en un MVR-tree.

Tabla 1. Lista de símbolos frecuentes

Símbolo	Descripción
$DIST_i$	Distribución de los objetos en el instante de tiempo i
T	Número total de todos los instantes de tiempo en la historia
f	Capacidad promedio de un nodo en un R-tree
K_i	Es el número total de nodos en el nivel i
E_i	Es la tasa de evolución en el nivel
b	Es la capacidad de objetos por nodo
P_{svo}	Versión fuerte del overflow para *b*
M_i	Es el número de nodos vivos en el nivel i

El modelo considera que en el instante de tiempo inicial (instante 1) los N objetos se encuentran distribuidos en un rango $[0,1]^d$ de acuerdo con una distribución que llamaremos DIST1.

Sea s_k el rango espacial correspondiente al rango limitado por todas las entradas *s*, y sea s_t el periodo de tiempo cuando *s* es válido. Entonces en una consulta $q(q_k,q_t)$, un nodo $s(s_k,s_t)$ debería ser visitado si y sólo si se intersectan q_k con s_k y q_t con s_t. En otras palabras, la probabilidad de que *s* sea visitado es idéntica a la probabilidad de que $s(s_k,s_t)$ se intersecte con $q(q_k,q_t)$, lo cual queda definido como $prob(s,q)$. Por otro lado, $prob_{espacial}$ y $prob_{temporal}$ denotan la probabilidad que las características y rangos temporales se intersecten. Es decir, considerando un espacio de dos dimensiones tenemos que:

$$prob(s,q)=prob_{espacial}\ prob_{temporal} \tag{1}$$

es decir,

$$prob_{espacial}=(s_1+q_1)(s_2+q_2) \tag{2}$$

donde

$$s_{i1} = s_{i2} = \sqrt{D_{i+1}\frac{f^{i+1}}{N}}\ (0\le i\le h\text{-}1) \tag{3}$$

donde

$$D_{i+1} = \left(1+\frac{\sqrt{D_i}-1}{\sqrt{f}}\right)^2 \text{ y } D_0=D \tag{4}$$

Donde D_i se refiere a la densidad de los MBR en el nivel *i*.

Por otro lado, la $prob_{temporal}$ está relacionada con el rango de evolución de los nodos. Sea P_i el número total de nodos en el nivel *i* cuyos ciclos de vida se intersectan con el nodo q_t y K_i el número de nodos creados en el nivel *i*, entonces tenemos:

$$prob_{temporal}=P_i/K_i \tag{5}$$

Basados en las fórmulas (3), (4) y (5) [3], el número de nodos accesados en un MVR-tree (ejemplo del costo en tiempo) en consultas del tipo *time-interval* y *time slice* es:

$$\sum_{i=0}^{\lceil \log_f N \rceil -1} \left\{ \frac{N}{f^{i+1}} \left(\sqrt{D_{i+1} \frac{f^{i+1}}{N}} + q_1 \right) \left(\sqrt{D_{i+1} \frac{f^{i+1}}{N}} + q_2 \right) \left[1 + \frac{a f_l^{i+1} \cdot (q_t - 1)}{\left(b - f_l^{i+1} \right)} \right] \right\} \tag{6}$$

Donde $\left[1 + \frac{a f_l^{i+1} \cdot (q_t - 1)}{\left(b - f_l^{i+1} \right)} \right]$ corresponde a la representación algebraica de E_i, donde *a* es la agilidad y *b* es la capacidad del nodo.

3. MVR-tree en Ambiente Paralelo

El escenario para la paralelización de consultas usando el índice espacio-temporal MVR-tree bajo el modelo CREW PRAM no dista mucho del modelo sobre un procesador. En [5] se postula que las consultas del tipo *time-interval* son más costosas que las consultas del tipo *time-slice* sobre un solo procesador. Por otro lado, en [5] sólo se presenta la estrategia local de distribución de los objetos. Es decir, se hace una división de N/P objetos para cada procesador, construyendo los índices respectivos en los P procesadores. No obstante, esta estrategia es una buena aproximación para enfrentar el problema. Por otro lado, se han propuesto otras estrategias paralelas como la distribución de árboles R-tree como estructura básica del MVR-tree [4]. No obstante, en [4] se argumenta que la paralelización de MVR-tree como estructura de datos, conlleva a un gran problema de balance de carga, ya que la estructura del MVR-tree es altamente cohesionada. Por ejemplo, en la Figura. 1.2. (b), podemos ver que R6' de N5 comparte a R4 que pertenece a N4. Esto permite suponer que en aplicaciones reales, en donde sólo un pequeño grupo de objetos cambian su estado o forma, sería extremadamente complicado paralelizar el MVR-tree por nodos (uno o más nodos raíces por procesador), ya que ésto implicaría altos costos de comunicación (habría que recorrer muchos nodos raíces entre los procesadores para responder la consulta).

En este artículo se propone otra distribución inicial de los objetos. Esto con el propósito de hacer una mejor distribución inicial de la carga y por ende disminuir la carga de trabajo para determinados procesadores. En esta estrategia se va asignando cada objeto i al procesador p. Posteriormente el objeto i+1 va al procesador p+1, el objeto i+2 va al procesador p+2, y así sucesivamente. Esto se realiza de manera circular cuando el número de objetos supera al número de procesadores.

3.1. Costo de consultas en un MVR-tree

La Ec. (6) obtiene el número de nodos accesados por una consulta de tipo *time-interval* en un procesador donde se almacena un MVR-tree parcial. Utilizando la Ec. (6) es posible obtener el costo total de nodos accedidos en el modelo CREW PRAM, el cual queda expresado en la Ec. (8) [5]. La Ec. (8) es una aplicación directa del paradigma *Tiempo de Trabajo* sobre modelos de computación paralela PRAM [9], el cual está expresado por:

$$W(n) = \sum_{i=1}^{S(n)} W_i(n)$$

Donde *W(n)* corresponde a la complejidad del trabajo y *S(n)* a la complejidad de los pasos paralelos para obtener la solución. Es por ello que reemplazando *W(n)* por $NA_{TotalParallel}(q)$ y *S(n)* por *P*, es posible obtener la Ec. (7)

$$\sum_{i=0}^{\lceil \log_2 N/P \rceil - 1} \left\{ \frac{N/P}{f^{i+1}} \left(\sqrt{D_{i+1} \frac{f^{i+1}}{N/P}} + q_1 \right) \left(\sqrt{D_{i+1} \frac{f^{i+1}}{N/P}} + q_2 \right) \left[1 + \frac{a f_l^{i+1} \cdot (q_l - 1)}{(b - f_l^{i+1})} \right] \right\} \tag{7}$$

$$NA_{TotalParallel}(q) = \sum_{p=1}^{P} NA_p(q) \tag{8}$$

3.2. Costo de consultas en función del tiempo

Sea $T(NA_i(q))$ el tiempo de procesamiento que le cuesta al procesador *i* para evaluar *q,* donde *q* es una consulta del tipo *time-interval* en un MVR-tree se tiene:

- $NA_i(q)$ Es el número de nodos accesados para una consulta *q* en un procesador *i.*
- $T(NA_i(q)) : N \rightarrow R^+, \ \forall NA_i(q) \geq 1$

Sea $T(NA_{parallel}(q))$ el tiempo de procesamiento paralelo para una consulta del tipo *time-interval* o *time-slice* en un MVR-tree en P. Donde:

- *P* es el número de procesadores.
- $T(NA_{parallel}(q)) = \max\{T(NA_1(q)),...,T(NA_p(q))\}$

Dado lo anterior y utilizando el teorema de Brent podemos estimar el tiempo de procesamiento de una consulta de tipo *time-interval* usando el MVR-tree en el modelo CREW PRAM con la Ec. (9) [5].

$$T(NA_{Parallel}(q)) = \max\{T(NA_1(q)),...,T(NA_p(q))\} + \log P \tag{9}$$

donde log *P* corresponde al costo de comunicación en un modelo CREW PRAM.

4. Evaluación Experimental

En esta sección se evalúan tres enfoques, la estrategia de distribución local, la estrategia de distribución circular, y la versión secuencial. La evaluación consiste en comparar las dos estrategias paralelas con la versión secuencial. Además, se evaluó el modelo asociado en [5] con respecto a las consultas del tipo *time-slice.*

Dado que no se cuenta con una máquina CREW PRAM, nosotros emulamos dicho modelo con un cluster de procesadores. Dicha simulación se realizó en un cluster de cuatro máquinas. El objetivo principal de estos experimentos es corroborar la disminución de la cantidad de nodos accesados en procesadores paralelos, lo cual trae como consecuencia una disminución del tiempo en el procesamiento de dichos nodos.

Las máquinas que componen el cluster son procesadores Pentium 4, 1.6 GHZ, 248 Mb RAM, HDD 60GB con sistema operativo Linux, Debian Sarge. En estos experimentos lo que se paralelizó

fue la versión del MVR-tree tal como debería de ser efectuada en un modelo paralelo CREW PRAM. La implementación utilizada es una versión libre del MVR-tree en C++ implementada por Yufei Tao. El lenguaje utilizado para realizar la implementación paralela fue MPI, utilizando la versión MPICH-1,

La paralelización se realizó 23.264 datos distribuidos secuencialmente sobre cuatro procesadores. El resto de los resultados experimentales con más de 4 procesadores fueron realizados por procesos. Se realizaron dos experimentos. El primer experimento consideró un 10% de movilidad, con 11 timestamp, se realizaron 46.520 inserciones en el MVR-tree y se ejecutaron 100 consultas del tipo *time-interval* y 100 consultas del tipo *time-slice.* Para las consultas del tipo *time-interval,* el intervalo considerado va desde 2 a 10 timestamp. Dichos timestamp fueron generados aleatoriamente.

Para el segundo experimento se consideró una movilidad del 20%, se consideraron 21 timestamp, se realizaron 116.113 inserciones en el árbol, y se ejecutaron 100 consultas del tipo *time-interval* y 100 consultas del tipo *time-slice.* Para las consultas del tipo *time-interval* se consideraron 10 timestamp fijos para cada consulta.

En la Figura. 2.1., podemos ver como el número total de nodos accesados va disminuyendo a medida que se van agregando procesadores y/o procesos a las consultas paralelas, esto con respecto a la versión secuencial y considerando las 100 consultas. La disminución de nodos accesados es tanto para la estrategia Local como la estrategia Circular.

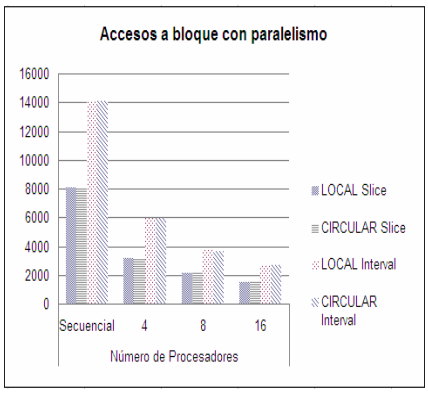

Figura 2.1: Número total de accesos a bloques para las 100 consultas, para las estrategias Local y Circular (Primer Experimento).

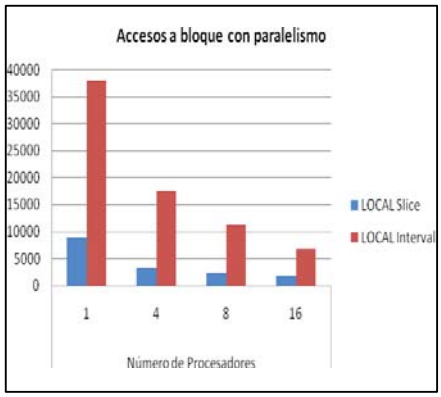

Figura 2.2: Ahorro promedio de número de nodos accesados para la estrategia Local (Segundo Experimento).

Figura 2: Número total de accesos a bloques y ahorro promedio de nodos accesados

También es posible apreciar en la Figura. 2.1., que las consultas del tipo *time-interval* para ambas estrategias entregan una mayor cantidad de nodos. En otras palabras las consultas del tipo *time-interval* son más caras tanto para la versión paralela como la secuencial. No obstante, la consulta del tipo *time-interval* es la que se ve mayormente beneficiada por el procesamiento en paralelo con respecto al tiempo que toma responder a dichas consultas.

Los porcentajes de ahorro con respecto a la versión secuencial son similares tanto para la estrategia Local como para la estrategia Circular. No existen diferencias significativas entre una y otra estrategia en cuanto al porcentaje de nodos accesados con respecto a la versión secuencial. Lo anterior, se debe a que todos los objetos en el espacio tienen la misma probabilidad de movimiento. Es por ello, que las dos estrategias entregan resultados equivalentes.

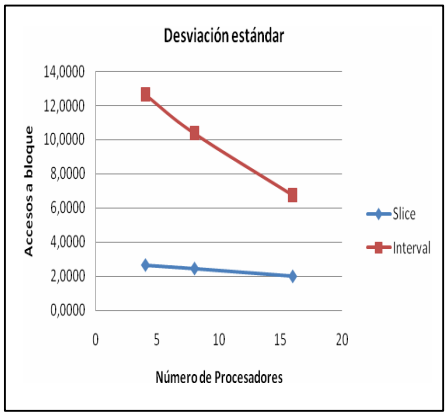

Figura 3.1: Desviación Estándar para las consultas de tipo time-interval y time-slice, para la estrategia Local (Segundo Experimento).

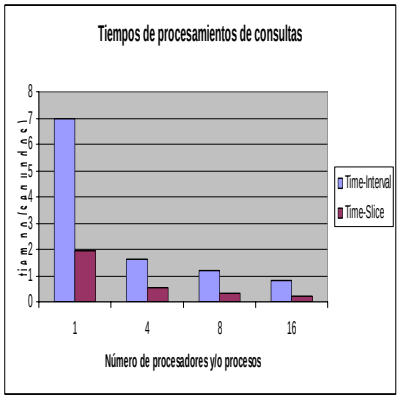

Figura 3.2: Tiempo de procesamiento para las consultas de tipo *time-interval* y *time-slice*, para la estrategia Local (Segundo Experimento).

Figura 3: Desviación estándar y tiempos de procesamiento de consultas

Debido a que las estrategias Local y Circular, tienen los mismos rendimientos, el segundo experimento sólo se realizó para la estrategia Local. Tras ello, es posible visualizar en la Figura. 2.2., la cantidad de nodos ahorrados para los dos tipos de consultas en paralelo con respecto a un solo procesador. Aquí es posible apreciar que la mayor cantidad de nodos ahorrados con respecto al número de procesadores y/o procesos, corresponde a las consultas del tipo *time-interval,* las cuales son más caras en cuando a la cantidad de nodos accesados en la versión secuencial.

En la Figura. 3.1., para el segundo experimento, es posible apreciar la desviación estándar con respecto al número de nodos accesados para las consultas del tipo *time-slice* y *time-interval* para la estrategia Local. Aquí es posible apreciar que la desviación estándar va disminuyendo con respecto al número de nodos accesados cuando se aumentan el número de procesadores y/o procesos. Es decir, la carga de trabajo es más equitativa cuando se aumenta el número de procesadores en ambas consultas. Más aún, podemos ver que la desviación estándar disminuye más rápidamente para las consultas del tipo *time-interval* en comparación con las consultas del tipo *time-slice* cuando se aumenta el número de procesadores.

Finalmente en la Figura. 3.2., es posible apreciar la disminución de tiempos de procesamiento en paralelo, con respecto a la versión secuencial. En la Figura. 3.2., se ve como disminuye el tiempo notablemente para consultas paralelas de 4, 8 y 16 procesadores y/o procesos. Generando un ahorro superior al 85% (en segundos) para 16 procesos con respecto a la versión secuencial.

4.1. Evaluación del modelo de costos paralelo

El propósito de esta sección es verificar la precisión del modelo de costos paralelo para la cantidad de nodos accesados en consultas del tipo *time-slice*. Con el propósito de realizar la evaluación, se tomaron los datos del experimento dos. Los resultados empíricos para consultas del tipo *time-slice,* se realizó sobre los cuatro procesadores reales. El resto de los experimentos para 8 y 16 procesadores se obtuvieron de procesos. Con el propósito de realizar la evaluación se han tomado los siguientes valores: D_0=1, con q_1=q_2=0,05, q_t=2, a=10%, P=4,8 y 16; y los siguientes valores P_{svo}=0,85, b=34 y f=23,56, para los símbolos considerados en la Tabla 1. Con dichos valores se ha obtenido un error relativo promedio de 55,92% ente el modelo de costos y los valores obtenidos empíricamente sobre 100 consultas de tipo *time-slice*. No obstante, la evaluación para las

consultas del tipo *time-interval* es más exacta mostrando un error promedio del 19,13% para 4, 8 y 16 procesadores [5].

5. Conclusiones y Trabajo Futuro

En este artículo se han presentado dos estrategias de paralelización para una máquina CREW PRAM. Dichas estrategias han utilizado el índice de acceso espacio-temporal MVR-tree. Ambas estrategias fueron comparadas con el enfoque secuencial (un solo procesador) demostrando ser muy eficientes, ya que existe un ahorro significativo de nodos accesados tanto para consultas del tipo *time-slice* como *time-interval*. Más aún, este ahorro se ve reflejado en los tiempos de procesamiento de ambas estrategias paralelas.

Otro punto relevante de destacar, es que no existen diferencias significativas entre el uso de la estrategia Local con respecto a la estrategia Circular, lo cual se debe a todos los objetos tienen la misma probabilidad de movimiento.

Por otro lado, es posible apreciar que los valores de desempeño para los dos experimentos no presentan diferencias significativas. Es decir, al momento de aumentar la agilidad de los cambios de los objetos, éstos no producen un cambio significativo en los costos de las consultas *time*-slice y *time-interval.* Más aún al dejar fijo los intervalos de las consultas del tipo *time-interval* en el segundo experimento, tampoco es posible apreciar una disminución de la eficiencia en las consultas paralelas.

También se realizó una verificación del modelo de costos paralelo del MVR-tree para las consultas del tipo *time-slice.* El modelo permite estimar el costo de las consultas con un de error promedio del 55,92%.

Como trabajo futuro se espera implementar otras estrategias de paralelización para un modelo de memoria distribuida.

REFERENCIAS

[1] Manolopoulos, Y., Theodoridis, Y., and J.Tsotras, V.: Advanced Database Indexing. 1st ed. Kluwer Academic Publishers. (1999)

[2] Gutiérrez G, Navarro G, Rodríguez A, and Gónzales A.: A Spatiotemporal Access Method based on Snapshots and Events. GIS'05. Bremen, Germany. (2005)

[3] Tao, Y., Papadias, D., and Zhang, J.: Cost models for overlapping and multiversion structures. ACM Trans. Database Syst. pp.299–342. (2002)

[4] Heum-Geun Kang and Chin-Wan Chung.: Exploiting Version for On-Line Data Maintenance in Molap Servers. Proceeding of the 28th VLDB conference. Hong Kong, China. (2002)

[5] Gutiérrez-Soto C., Gutiérrez G., Rodríguez P., Campos P.,: Paralelización de Consultas del tipo time-interval en Base de Datos Espacio-Temporales. JIISIC08. Guayaquil, Ecuador. (2008)

[6] Bayer R., and McCreight E..: Organization and maintenance of large ordered indices. Acta Inf.. Volume 1.pp. 173–189. (1972)

[7] Guttman A.: R-trees: A dynamic index structure for spatial searchinng. In ACM SIGMOD Conference on Management of Data. ACM. pp. 47–57. (1984)

[8] Papadias, D., Kalnis, P., Zhang, J., Tao, Y. Efficient OLAP Operations in Spatial Data Warehouses. In procceding of the International Symposium on Spatial and Temporal Databases (SSTD) (July). pp. 443-459. (2001)

[9] Chatterjee, S., Prins, J.,: COMP 203: Parallel and Distributed Computing PRAM Algorithms. Spring. (2002)