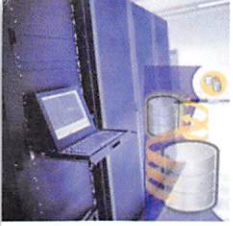


Construcción de escenarios para comparar la replicación activa optimista y pesimista

Edison Álvarez
Ingeniero en Sistemas, Docente de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial

RESUMEN

En la actualidad debido al gran volumen de información que se procesa y no en un solo lugar, es necesario determinar los escenarios más adecuados para aplicar y configurar la distribución de la información en las organizaciones. Los Administradores de Base de Datos distribuyen la información entre varios sitios, pero no configuran los escenarios acorde a las necesidades reales de rendimiento, seguridad y disponibilidad de la información.

El presente trabajo pretende mostrar los elementos que se deberían considerar para configurar escenarios en los cuales se aproveche las ventajas de distribuir la información, no solo considerando aspectos de seguridad sino también factores que mejoren sustancialmente el rendimiento de los sistemas de base de datos.

Se estudia dos técnicas de replicación, la Replicación Activa, con sus variantes pesimista y optimista, y la Replicación Pasiva. Se establece el marco de trabajo de cualquier técnica de replicación en 5 fases: Solicitud del cliente, Coordinación de Servidores, Ejecución, Coordinación de Asentimientos y Respuesta al cliente. Se explora este marco en el contexto de la Replicación Activa y Pasiva.

Finalmente se compara la replicación Activa Pesimista y la Replicación Activa Optimista (que en SQL Server corresponden a la replicación Transaccional con actualización inmediata y actualización en cola, respectivamente). Para lograr este objetivo se construyó un Front - End que actúa como un cliente que realiza solicitudes de ejecución de transacciones tanto de consulta como de actualización, se configuró además los escenarios de replicación y realizaron experimentos variando una serie de parámetros, con lo que se obtuvo datos estadísticos que posibilitaron realizar una comparación entre las técnicas de replicación antes mencionadas y determinar ventajas y desventajas.

ABSTRACT

At present due to the large volume of information being processed and not in one place, it is necessary to determine the most appropriate settings to implement and configure the distribution of information in organizations. The Database Managers distribute the information among multiple sites, but do not configure the settings according to the real needs of performance, security and availability of information.

This paper aims to show the items that should be considered to set up scenarios which take advantage of distributing information, not only considering safety aspects but also factors that substantially improve the performance of database systems.

Transactional replication is selected for a more detailed study, selection justified from the point of view that this type of replication ensures transactional consistency, a condition required in most commercial applications, relying on a protocol that does not belong to replication itself, and that is the Commit Protocol in two phases, however, also addresses two other ways to replicate information.

Finally, we compare the replication Turns Pessimistic and Optimistic Active Replication (which correspond to SQL Server transactional replication with immediate updating and queued updating, respectively). To achieve this goal we built a Front - End acts as a client that makes requests of execution of both query transactions and update, it also set up replication scenarios and conducted a series of experiments varying parameters, thereby statistical data obtained enabled a comparison between replication techniques mentioned above and identify advantages and disadvantages.

INTRODUCCIÓN

Un área en la cual las soluciones están integrando tecnología con nuevas arquitecturas o formas de hacer las cosas es, sin lugar a dudas, el área de los *sistemas distribuidos de información*. Ellos se refieren al manejo de datos almacenados en facilidades de cómputo localizadas en muchos sitios conectados a través de una red de comunicaciones. Un caso específico de estos sistemas distribuidos es lo que se conoce como *bases de datos distribuidas*, donde la posibilidad de distribución se la realiza a través de la *replicación* y en otros casos a través de la fragmentación de la información, desde luego sin olvidar la posibilidad de una distribución combinada.

Una **base de datos distribuida** (BDD) es un conjunto de múltiples bases de datos *lógicamente relacionadas*, las cuales se encuentran distribuidas entre diferentes sitios interconectados por una red de comunicaciones.La réplica permite distribuir de forma automática copias de los datos de un servidor a uno o varios servidores de destino en uno o varios emplazamientos remotos [Soukup98].

Una base de datos replicada consiste de un grupo de sitios $n = \{N1, N2, N3, ..., Nn\}$ los cuales se comunican por el intercambio de mensajes. Los sitios son detenidos al fallar, y las fallas en los sitios pueden ser detectadas. Se considera un modelo de recuperación de fallas en el cual los sitios pueden recuperarse y reconectarse al sistema después de sincronizar su estado con alguna de las réplicas que estén corriendo. La base de datos es replicada totalmente, es decir, cada sitio contiene una copia de la base de datos.

Los clientes interactúan con la base de datos a través de las transacciones. Las transacciones son ejecutadas automáticamente y éstas pueden ser confirmadas o abortadas en todos los sitios. Las transacciones son parcialmente ordenadas en conjunto de operaciones de lectura (**r**) o de escritura (**w**). Si una transacción posee operacio-

nes de escritura, un protocolo de confirmación en dos fases (2PC por sus siglas en inglés, TwoPhaseCommit) es ejecutado al final de la transacción entre todos los sitios. En las bases de datos replicadas, el criterio correcto es una copia serializable, es decir, cada copia debería aparecer como una simple copia lógica y la ejecución de la actual transacción será equivalente a una ejecución en serie sobre todas las copias físicas.

Cuando cada réplica está corriendo en máquinas diferentes, los sistemas replicados de base de datos tiene, en teoría, dos ventajas sobre los sistemas centralizados:

- Alta disponibilidad.- si una réplica colapsa[2] (*crash*), sea por una falla de hardware o software, el resto de réplicas pueden continuar operando.
- Mejor rendimiento.- la carga del procesamiento transaccional puede ser distribuida entre todas las réplicas (máquinas) en el sistema. Esto contribuye a:

  - Mayor salida (throughput): las réplicas pueden independientemente ejecutar *consultas* y las operaciones de *lectura* de las transacciones de actualización, porque ellas no alteran el estado de la base de datos, es decir, el procesamiento de un mayor número de transacciones por unidad de tiempo.
  - Menores tiempos de respuesta: ya que las consultas pueden ser ejecutados sobre una réplica y enviar la respuesta al cliente, sin comunicaciones adicionales entre las réplicas.

Por otra parte, las ventajas antes mencionadas, generan los siguientes costos:

- Procesamiento adicional y overheaden las comunicaciones: las réplicas requieren comunicarse para asegurase que los cambios han tomado efecto en todas las copias de base de datos. Esto incrementa la carga en las máquinas(más precisamente en el subsistema de comunicaciones) y las comunicaciones de red, las cuales pueden degradar todo el rendimiento.
- Sistemas altamente complejos: las réplicas corren asincrónicamente[3] sobre diferentes máquinas y asincrónicamente reciben las solicitudes de los clientes para modificar las bases de datos. Realmente sincronizar las copias de las bases de datos a través de las réplicas requiere de algoritmos avanzados de comunicación y procesamiento de transacciones.

METODOLOGÍA

Para realizar el estudio comparativo entre la Replicación Activa Optimista y la Pesimista se requiere:

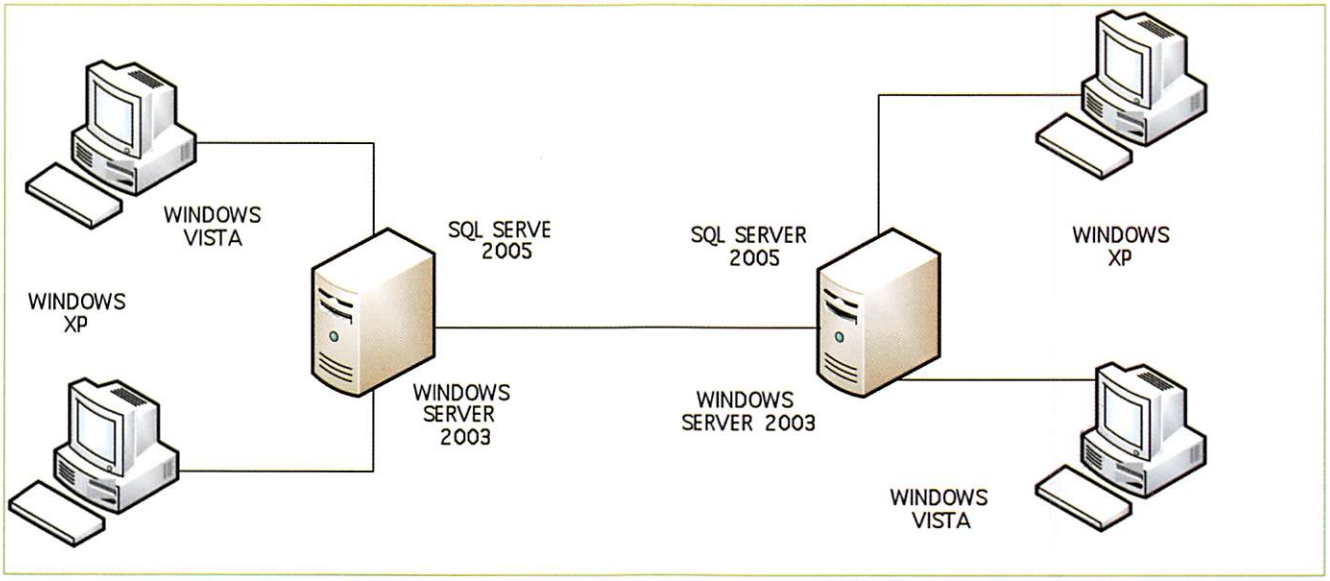
a) Definir escenarios de replicación activa optimista y pesimista
b) Construir un prototipo de replicación
   a. Establecer los parámetros de comparación
   b. Número de operaciones por transacciones
   c. Número de operaciones de lectura y número de operaciones de escritura
   d. Condiciones para anular una transacción (Rollback)
   e. Porcentaje de transacciones a ser confirmadas (Commit)
   f. Tamaño de la base de datos (tipos de datos a emplear y número de registros a procesar)
c) Construcción de un Front – End para la obtención de los valores de medición de los parámetros.
d) Elaboración de las gráficas en base a los resultados obtenidos
e) Interpretación de resultados.

a) Escenario de Replicación

El escenario construido, es el que se muestra en la figura siguiente, cada estación tam-

[2] Es decir, un proceso deja de funcionar definitivamente, según la definición encontrada en [Vanderwall2000]
[3] Realmente las réplicas son sincrónicas y asincrónicas

bién incluye a SQL Server 2005, cada servidor es a su vez Publicador y Distribuidor.



b) Prototipo de Replicación

Partiendo del hecho que la Replicación Activa no permite modificaciones independientes en las réplicas, es decir, mantiene la consistencia de los datos replicados en todas ellas, se puede considerar que la Replicación Activa Pesimista corresponde al modelo de Consistencia Estricta – Replicación de actualización inmediata de subscripciones, y la Replicación Activa Optimista al modelo de Consistencia No Estricta donde se permite que exista un intervalo de tiempo no nulo entre los cambios que se producen en los datos originales y su propagación a las demás copias, es decir, habilita a los suscriptores para actualizaciones en cola, las modificaciones de los datos pueden ser hechas en el Suscriptor, almacenarlas en una cola y propagarlas al Publicador. Ambos modelos corresponden a la replicación transaccional donde se mantiene la consistencia e integridad transaccional.

Parámetros del prototipo

Número de operaciones por transacción: >= 5.
Esto significa que cada transacción podría contener 3 operaciones de datos, modeladas de la siguiente manera: 1 operaciones de **lectura**, 1 operaciones de **escritura** y 1 de **actualización** o una combinación de ellas. Una operación es Iniciar transacción y otra es una operación de confirmación (Commit) o una operación de deshacer (Rollback).

Porcentaje de transacciones confirmadas: 100.

Número de transacciones solicitadas por unidad de tiempo: *@@Lock_timeout*

Por otra parte, el número de transacciones solicitadas está relacionado con el *"throughput" (salida)* del sistema: el número de transacciones que son confirmadas por unidad de tiempo. Sea *Cr* el conjunto de transacciones confirmadas en una corrida *r*, entonces definimos:

$$throughput(r) = \#Cr$$

#C es la cardinalidad del conjunto *C*, por ejemplo, el número de elementos que *C* contiene.

Tamaño de la base de datos: 50000 registros.

Tamaño de los objetos de la base datos: 1 – 30 bytes.

Técnicas de replicación: Replicación Activa Pesimista y Replicación Activa Optimista.

Número de réplicas: 2 a 4.

Interactividad de las transacciones: Interactivas, es decir, cada cliente solicita transacciones una a continuación de otra.

Porcentaje de consultas: 0 a99.

Número de clientes: 1 a 8.

Escenarios: sea el escenario s = (*pesimista*;3; *interactivas*;50%;8), significa **s** es un escenario en el cual la técnica de replicación pesimista es usada, en un sistema con 3 réplicas. Las transacciones son interactivas y el 50% de ellas son consultas y el número total de clientes en el sistema son 8.

Experimento: sea un experimento *e* definido como *e* = ({*pesimista*; *optimista*};{2;3;4}; *interactivas*;{0%à99%};5).

Indicadores de Rendimiento

Tiempo de respuesta medio por transacciones confirmadas y el *throughput*.

proc(o) La cantidad de tiempo que demora *o* desde el momento en que *o* es solicitada por el cliente, hasta el momento en que el cliente recibió el resultado de *o*.

net(o) La cantidad de tiempo que demora *o* en el componente de Comunicación (red) desde el momento en que *o* es solicitada por el cliente, hasta el momento en que el cliente recibió el resultado de *o*.

total(o) La cantidad de tiempo entre el momento que *o* es solicitada por el cliente, hasta el momento en que el cliente recibió el resultado de *o*.

Este valor es igual a: *proc(o)* + *net(o)*.

Tiempo de respuesta medio para las transacciones confirmadas

$$total(T) = \sum_{o \in O} total(o)$$

Sea *CR* el conjunto que contiene todas las transacciones confirmadas en un conjunto de corridas *R*. El indicador de rendimiento tiempo de respuesta medio para las transacciones confirmadas se define como:

$$medio(R) = \frac{\sum_{T \in C_R} total(T)}{\#C_R}$$

Sea *#transacciones*(r) el número de transacciones en una corrida r, entonces:

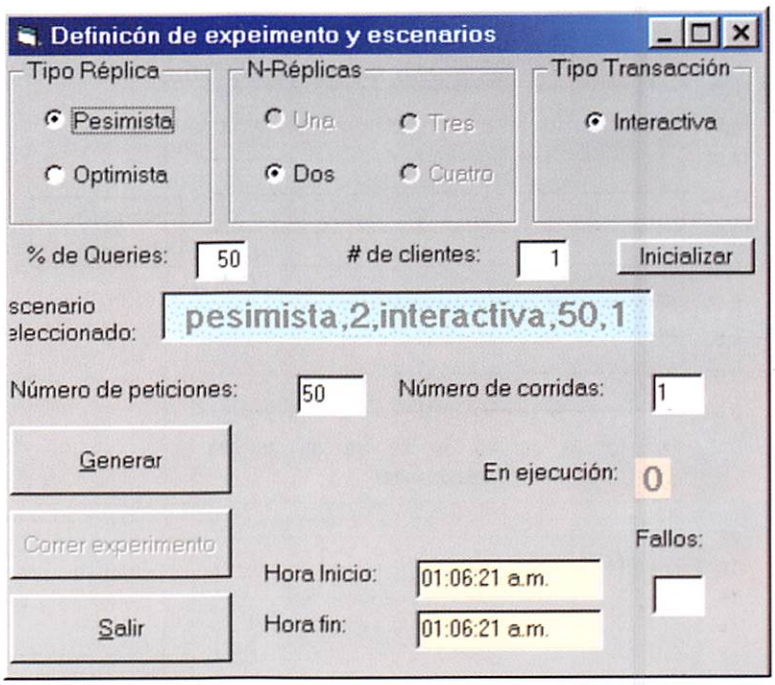
$$throughput(r) = \#transacciones(r) \,/\, media(r)$$

Sea *R* el conjunto de corridas. *media*(*R*/*c*) denota el tiempo de respuesta medio para las transacciones de consultas confirmadas en R. *media*(*R*/*a*) es el tiempo medio de respuesta para las transacciones de actualización confirmadas en R.

$$media(R) = (media(R/c) + media(R/a)) \,/\, n$$

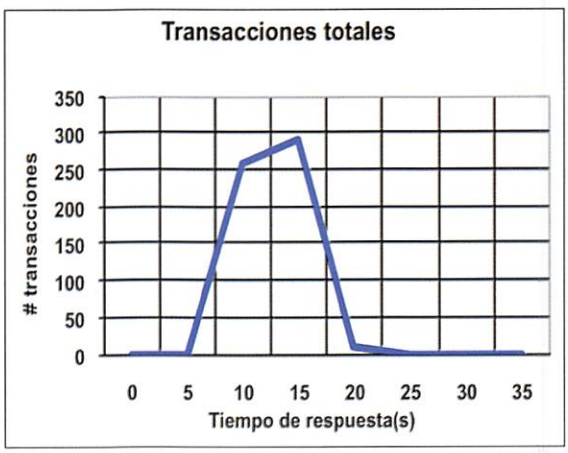
c) Construcción del Front – End

Para este estudio se programó una aplicación básica que permite construir el escoger el escenario a aplicar en base a: Tipo de Réplica, Número de réplicas, Tipo de Transacción (en nuestro caso no se consideran transacciones de solo lectura – read only), el Porcentaje de Consultas (Queries), Número de clientes participantes, el Número de peticiones y el número de corridas de prueba. La siguiente figura muestra la interfaz de la aplicación desarrollada.

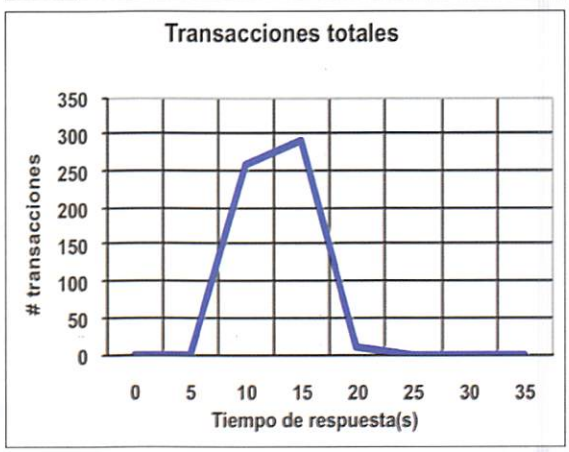


Elaboración de gráficas según los resultados obtenidos. (Comparativa)

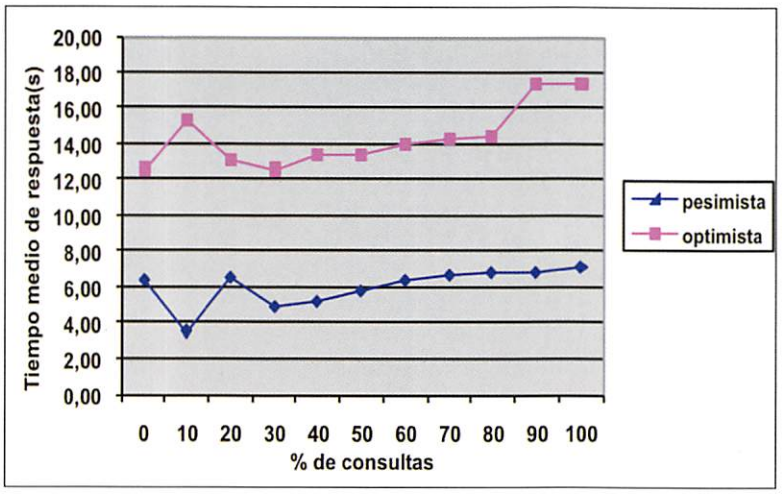
Distribución de frecuencias de tiempos de respuesta para la replicación pesimista



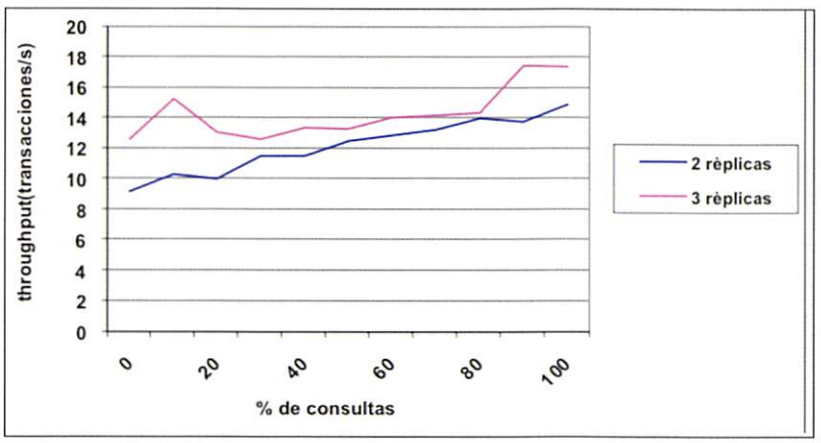
Distribución de frecuencias de tiempos de respuesta para la replicación optimista



Tiempos de respuesta medio por técnica variando el porcentaje de consultas

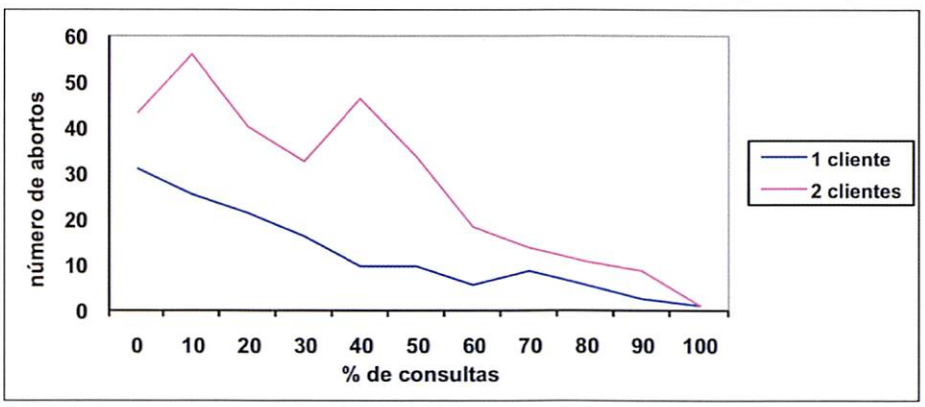


Throughput de la replicación pesimista variando el número de réplicas

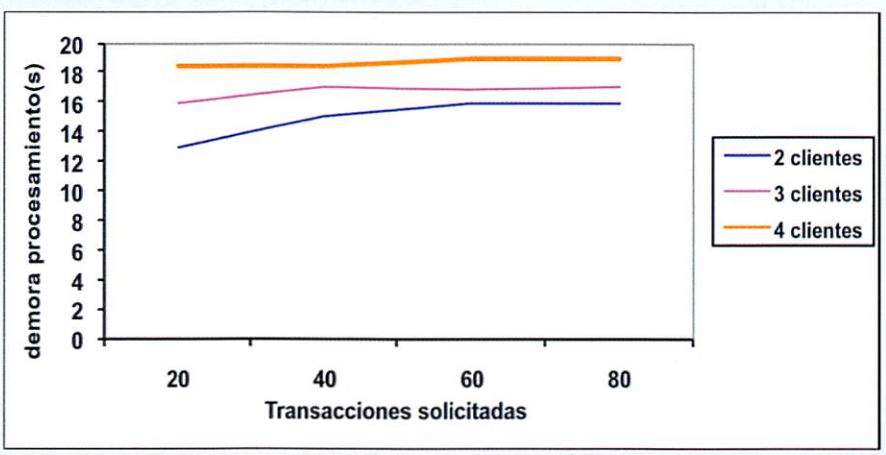


Throughput de la replicación optimista variando el número de réplicas

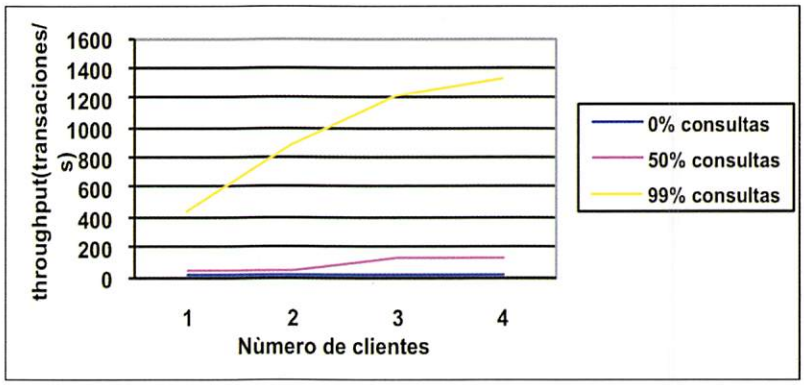
Abortos variando el porcentaje de consulta



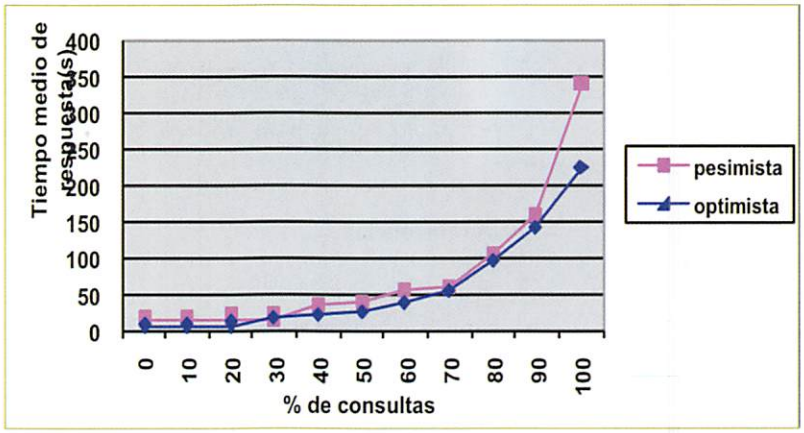
Evolución de la demora en el procesamiento



Throughput en función del número de clientes



Tiempos de respuesta medio por técnica variando el porcentaje de consultas



CONCLUSIONES

- La replicación de instantáneas es útil para suscriptores que solo requieren datos de sólo lectura. Pero no es adecuada cuando el volumen de datos es enorme, los suscriptores requieren continuamente datos actualizados y necesitan realizar sus propias actualizaciones, es decir, el rendimiento decrece a medida que se incrementa el volumen de peticiones.

- El número de clientes en el escenario propuesto, determina el grado de concurrencia, considerando que las transacciones son interactivas y se producen solicitudes de ejecución cada cierto intervalo de tiempo, lo que afecta al throughput del sistema. Mientras más clientes participen mayor será el tiempo de demora. Se aplica el criterio de serialización.

- El throughput del sistema disminuye al aumentar el número de replicas participantes, sin embargo se ve mejorada al utilizar la técnica de replicación optimista. Se evita la propagación de la replicación a través de una sola copia primaria.

- Al configurar adecuadamente los catálogos el rendimiento mejora significativamente.

BIBLIOGRAFÍA

- [Vandewall2000]R. Vandewall, Database Replication Prototype, Department of mathematics Computer Science University of Groningen, Groningen ,Netherlands
- [Jimenez2001]R. Jimenez-Peris, M. Patino-Martinez, G. Alonso, B. Kemme, How to Select a Replication Protocol According to Scalability, Availability and Communication Overhead , 2001
- [Patiño2002]M. Patiño-Martínez, R. Jiménez-Peris, B. Kemme, G. Alonso: Scalable

Replication in Database Cluster In: 14th International Symposium on Distributed Computing (DISC), Toledo, Spain, October 2002.

- [Kemme2000]B. Kemme, G. Alonso: Don't be lazy, be consistent: Postgres-R, A new way to implement Database Replication. In: 26th International Conference on Very Large Databases (VLDB), Cairo, Egypt, September 2000
- [Pedone2000]M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, G. Alonso: Understanding Replication in Databases and Distributed Systems. In: 20th International Conference on Distributed Computing Systems (ICDCS), Taipei, Taiwan, Republic of China, April 2000.
- [Salas06] Jorge Salas, Lightweight Reflection for Middleware-based Database Replication, Universidad Politécnica de Madrid, 2006
- [Coulouris04]Coulouris George, Dollimore Jean, and Kindberg Tim, Distributed Systems Concepts and Design, Fourth ed., Addison Wesley, USA, 2004
- [Tanenbaum08]Andrew S. Tanenbaum, and Maarten Van Steen, Sistemas distribuidos. Principios y paradigmas, 2nd ed., Pearson Pretince Hall, Estado de México, México, 2008
- [Soujov06] Sujoy Paul, Pro SQL Server 2005 Replication, Apress, 2006

Otras referencias

- http://msdn.microsoft.com/es-es/library/ms151198.aspx
- http://social.msdn.microsoft.com/Forums/es-ES/sqlserveres/thread/c5fbdbf8-f38b-4afe-bbad-8f935e2f6379
- http://technet.microsoft.com/es-es/library/ms151799.aspx
- http://support.microsoft.com/kb/142800/es
- http://www.newsgrupos.com/microsoft-public-es-sqlserver/337709-replicacion-de-base-de-datos-sql-2005-a.html