Simulador para el estudio de casos de procesamiento distribuido en tiempo real.

Ivana Miatón
Patricia Pesado
Armando E. De Giusti

L.I.D.I., Facultad de Informática, Universidad Nacional de La Plata
La Plata, Argentina, 1900
{imiaton,ppesado,degiusti }@lidi.info.unlp.edu.ar

Resumen

Dado que un sistema distribuído de tiempo real debe interactuar con el mundo real, en puntos fisícamente distantes y en períodos de tiempo que vienen determinados por el contexto o las restricciones de la especificación (en muchos casos a partir de una activación asincrónica), es necesario y útil trabajar en aspectos de planificación, desarrollo y verificación de software para este tipo de sistemas.

El objetivo de este trabajo es presentar un simulador para estudiar, monitorear, medir y comparar tiempos de respuesta al ejecutar transacciones distribuídas (concurrentes o no) considerando la posibilidad de fallos en cualquiera de las localidades involucradas en la misma.

Se presentan aquí los resultados de haber utilizado el ambiente en la simulación del mantenimiento y recuperación de datos en un sistema de bases de datos distribuídas, en el que se plantean problemas de concurrencia de procesos en BDD con replicación y el aseguramiento de la integridad de las transacciones. El modelo de simulación permite estudiar el comportamiento de una base de datos distribuída incluyendo la simulación de fallos sobre la misma y la performance para recuperación utilizando el protocolo de dos fases.

El soporte de hardware del sistema es una red LAN-WAN y el ambiente ha sido desarrollado en JAVA.

KEYWORDS: Sistemas distribuidos.Tiempo real. Bases de Datos distribuidas. Replicaciòn de datos y procesos.

1. INTRODUCCION

Un sistema distribuido consiste en un conjunto de computadoras autónomas conectadas por una red y con soporte de software distribuido. Las computadoras pueden coordinar sus actividades y compartir recursos de hardware, software y datos, de manera tal que el usuario percibe una única facilidad de cómputo integrada aunque esta pueda estar implementada por varias máquinas en distintas ubicaciones [COU95].

Sintéticamente algunas ventajas del procesamiento distribuido son [GOM97]:

- ✓ Mejora de la disponibilidad: la operación es factible en una configuración reducida cuando algunos nodos están temporalmente no disponibles. No hay un punto centralizado y único de falla.
- ✓ Configuración más flexible: una aplicación puede configurarse de distintas maneras, seleccionando el número apropiado de nodos para una instancia dada.
- ✓ Control y administración más localizada: un subsistema distribuido, ejecutando en su propio nodo, puede diseñarse para ser autónomo, de modo que puede ejecutar en relativa independencia de otros subsistemas en otros nodos.
- ✓ Expansión incremental del sistema: si existe sobrecarga, el sistema puede expandirse agregando más nodos.
- ✓ Balance de carga: en muchas aplicaciones la carga total del sistema puede ser compartida entre varios nodos.
- ✓ Mejora en el tiempo de respuesta: los usuarios locales en nodos locales pueden obtener respuestas más rápidas a sus requerimientos, que accediendo a un ùnico servidor remoto.

En particular un sistema distribuido de tiempo real (SDTR) debe interactuar con el mundo real, en puntos físicamente distantes y no necesariamente fijos, en períodos de tiempo que vienen determinados por el contexto o las restricciones de la especificación (en muchos casos a partir de una activación asincrónica). [HAT88]. Naturalmente esto incrementa las dificultades del desarrollo de software para SDTR, entre las cuales pueden mencionarse [SHU92]:

- ▪ Manejo de mensajes asincrónicos con diferente prioridad.
- ▪ Detectar y controlar condiciones de falla, a nivel de software, de procesadores y de comunicaciones. Prever diferentes grados de recuperación del sistema.
- ▪ Modelizar condiciones de concurrencia y paralelismo.
- ▪ Manejar las comunicaciones inter-procesos e inter-procesadores.
- ▪ Asegurar la confiabilidad de los datos y analizar su migración en condiciones de funcionamiento normal o de falla. En casos de falla fi sica puede ser necesaria la migraciòn de procesos en tiempo real.
- ▪ Optimizar la redundancia de software y el grado de replicaciòn de datos para tener mejores tiempos de repuesta y mayor confiabilidad global.
- ▪ Organizar y despachar la atención de procesos, manejando las restricciones de tiempo especificadas.
- ▪ Testear y poner a punto un sistema físicamente distribuido.

Todas estas dificultades conducen a la utilidad de desarrollar ambientes de experimentación que permitan modelizar el sistema distribuido y simular condiciones de funcionamiento real, de modo de verificar las especificaciones o estudiar tiempos de respuesta.

El aporte de este trabajo, que es una evolución de la Tesina de Licenciatura en Informática de la Lic. Miaton es el desarrollo de un ambiente experimental en JAVA para estudiar condiciones de funcionamiento real (fallas de nodos, replicación de datos, migración de datos y procesos, etc) de sistemas distribuidos sobre una red LAN. [MIA01]

2. BASES DE DATOS DISTRIBUIDAS

Dado que gran parte del desarrollo experimental se ha hecho hasta el momento sobre casos de Bases de Datos distribuidas que se procesan en tiempo real, a continuación se resumen algunos aspectos de importancia en BDD [BELL92] [IEEE] [VAL99] [ZAN97]

El modelo distribuido de datos hace posible la integración de BD heterogéneas proveyendo una independencia global del administrador de bases de datos (DBMS) respecto del esquema conceptual. Además, es posible implementar una integración tal que reúna varios modelos de datos, representando cada uno de ellos características propias de organizaciones diferentes, asociadas para un trabajo conjunto.
En nuestro trabajo hemos supuesto un soporte multiprocesador MIMD (Múltiple Instruction, Multiple Datastream), en el cual interesan los problemas de distribución óptima de datos y procesos, de migración de datos y procesos y de tolerancia a fallas.

Algunos temas de particular interés que se pueden experimentar con el ambiente desarrollado son: Tasa de pérdida de datos en condiciones de falla; Tiempo máximo necesario para recuperación de información; Complejidad y eficiencia de los algoritmos de recuperación; Tiempo de utilización de recursos del sistema; Incidencia del porcentaje de replicación en el tiempo de respuesta; Grado de replicación óptimo para una clase de aplicación; Eficiencia de los algoritmos de migración de datos y procesos.

Algunos conceptos en BDD

- ✓ La *integridad de datos* se refiere a la capacidad de las bases de datos de manejar actualizaciones concurrentes de datos que están en varias ubicaciones físicas y de asegurar que todos ellos sean física y lógicamente correctos.
  La *recuperación* en una BDD significa que la transacción en su totalidad sea completada de manera exitosa.

- ✓ En un ambiente de bases de datos distribuidas, *las transacciones pueden acceder a datos almacenados en más de un lugar*. Cada transacción es dividida en un número de sub-transacciones, una por cada lugar en donde los datos accedidos por al transacción están almacenados. Estas sub-transacciones están representadas por *agentes* en los distintos sitios.

- ✓ La *indivisibilidad* de toda la transacción global es fundamental, pero además cada sub-transacción (agente) de la transacción global debe ser tratada como una transacción indivisible en el sitio donde está ejecutándose. Las sub-transacciones de la transacción global no sólo deben ser sincronizadas con otras transacciones concurrentes locales, sino que además deben sincronizarse con otras transacciones globales que se ejecutan concurrentemente en el sistema.

- ✓ En los sistemas de bases de datos distribuidas para asegurar la consistencia ante fallas, es necesario desarrollar algoritmos de recuperación de transacciones, que resultan complejos pues se necesita *atomicidad* tanto para las sub-transacciones locales como para las transacciones globales.

- ✓ Debe evitarse el *deadlock* de los sitios debidos a fallas en otros sitios o de comunicaciones. Es importante el protocolo de commit que utilice el administrador de la BDD para mantener la consistencia de la transacción global [BELL92]. En este trabajo se utiliza el protocolo de dos fases [VAL99].

- ✓ En BDD se puede utilizar un cierto grado de *replicación* de los datos buscando mejorar la confiabilidad y maximizar la velocidad de acceso por un incremento de la localidad de los datos. Si bien es difícil generalizar el grado óptimo de replicación para la arquitectura del sistema, puede estudiarse con bastante precisión el problema para una dada aplicación.

3. AMBIENTE DE SIMULACION

El ambiente de simulación fue implementado en Java. Se utilizó el JDK 1.3 junto con el Forte for Java 2.0 para construir la interfase con el usuario (GUI) [JAVA00] [JAVAW1]. De esta manera, cualquier computadora que posea el JVM correspondiente, podrá participar de las simulaciones.

La simulación comienza desde una máquina iniciadora donde se encuentra la clase *Manejador*. Allí se especifican cuales serán las máquinas involucradas (a través de sus direcciones IP), las transacciones que se quieren ejecutar, dónde se desea que se inicie cada una de ellas (qué sitio la coordinará) y qué máquinas tendrán fallas y en qué momento.

La máquina iniciadora no participa de la simulación, para no interferir en el funcionamiento de los que serán participantes de la base de datos distribuida.

Una vez que se inicia la simulación se le envía dicha información a los sitios correspondientes. (Cabe destacar que cuando se habla de máquina y sitio, se hace referencia a lo mismo).

En cada sitio participante de la simulación debe estar corriendo un proceso llamado *ProcesadorDeServidor*. Este se encarga de administrar lo que sucede en la máquina donde se encuentra, con un alcance local.

El funcionamiento de un servidor implica la escucha de conexiones, su aceptación, el procesamiento de solicitudes por las conexiones y su finalización una vez que todas las

solicitudes hayan sido procesadas. El manejo de conexiones múltiples se ejecuta por medio de múltiples threads. Por lo tanto, como el *ProcesadorDeServidor* puede llegar a recibir peticiones de ejecución de varios procesos (pedidos para ejecutar algún Coordinador de transacciones globales, y Participantes de otras transacciones globales), y para que no hayan esperas, se crea otro proceso (Thread) llamado *ClienteDeServidor* que se encarga de procesar uno de esos pedidos (existe uno por pedido), logrando que puedan satisfacerse varios al mismo tiempo.

De esta manera, el *ProcesadorDeServidor* está siempre dispuesto a recibir peticiones para ejecutar consultas, y cuando recibe una, crea un *ClienteDeServidor* para que se encargue de ella. Si recibe un pedido del *Manejador*, normalmente es para que se encargue de coordinar una transacción global. En este caso se crea un proceso *Coordinador* y se le da comienzo. Si recibe un pedido de un *Coordinador*, le pedirá que haya un agente que se encargue de ejecutar una sub-transacción. En este caso se creará un proceso *Localidad* que se encargará de realizar dicha tarea. Y también le dará comienzo.

En cada sitio existe un mapa que indica en qué lugares se encuentran las distintas tablas de la base de datos, ya que como existe replicación de datos, y se está utilizando la actualización sincrónica, deben actualizarse todas las bases de datos al mismo tiempo. Así, el *Coordinador* de una transacción sabe a quién tiene que pedir la ejecución de una sub-transacción.

Para la ejecución de las transacciones globales se sigue el protocolo de Commit de dos Fases (2PC). La utilización de este protocolo implica una gran comunicación entre el *Coordinador* y los *Participantes* de una transacción global, para saber si se comete efectivamente o si hay que abortar para que la base de datos quede en un estado consistente.

Esta comunicación entre procesos se realiza mediante sockets [TCPW1] [JAVAW2]. En la actualidad, la interface socket constituye el método más usado para el acceso a una red TCP/IP. Un socket es una abstracción que representa un enlace punto a punto entre dos programas ejecutándose sobre una red TCP/IP. Utiliza el modelo Cliente/Servidor. Cuando dos computadoras desean comunicarse, cada una usa un socket. Una de ellas es el "Server" que abre el socket y espera por alguna conexión. La otra es el "Cliente" que llamará al socket server para iniciarla. Además, para establecer la conexión, solo es necesaria la dirección del Server y el número de puerto que se utilizará para la comunicación. En este trabajo, se utiliza el package java.net. Cuando los dos sockets están comunicados, el intercambio de datos se realiza mediante InputStreams y OutputStreams.

4. EJEMPLOS DE UTILIZACION DEL AMBIENTE

Las simulaciones efectuadas se realizaron sobre una LAN-WAN. De esta manera cada terminal simula una base de datos que es parte de una base de datos distribuida. Y las comunicaciones se realizan a través de los mencionados sockets. A continuación se presentan resultados sobre un número limitado de sitios (máquinas) y se está trabajando en un reporte donde se emplean 8 y 16 sitios, homogeneos y heterogéneos.

Caso 1:

El ambiente simulado es de 3 máquinas que se llamarán X, Y, Z. En todas ellas existen réplicas de las tablas UNO, DOS y TRES. El Manejador se encuentra en una cuarta máquina para no interferir en el funcionamiento de los que serán participantes de la base de datos distribuida.
La tabla de tiempos de ejecución, medidas en milisegundos, muestra en cada máquina cuánto tiempo demoraron los distintos procesos que se ejecutaron. Con esta tabla se pueden obtener varias conclusiones.

1. Ejecución de una transacción sobre UNO coordinándose en X.
2. Ejecución de una transacción sobre DOS coordinándose en Y.
3. Ejecución de una transacción sobre TRES coordinándose en Z.

1.1: Falla simulada: Sin fallas

Tablas de tiempos de ejecución:

Máquina X			Máquina Y			Máquina Z		
Activ.	Consulta	T(ms)	Activ.	Consulta	T(ms)	Activ.	Consulta	T(ms)
Coord.	1	4500	Coord.	2	4500	Coord.	3	4340
Part.	1	3730	Part.	1	3400	Part.	1	3840
Part.	2	2410	Part.	2	2800	Part.	2	2750
Part.	3	3350	Part.	3	2690	Part.	3	3350

El que se presentó es el caso de la ejecución normal de transacciones concurrentes en la base de datos distribuida, con replicación de datos. Las tres transacciones involucradas se completan satisfactoriamente.
El tiempo que tarda en completarse la ejecución global, se ve representado en el tiempo de vida del coordinador. Cabe destacar que algunas diferencias en los tiempos dependen de las características físicas de las máquinas utilizadas.

1.2: Falla simulada: Caída de X antes de que el Participante de la consulta 2 haya recibido el mensaje de “preparar” del Coordinador.

Tablas de tiempos de ejecución:

Máquina X			Máquina Y			Máquina Z		
Activ.	Consulta	T(ms)	Activ.	Consulta	T(ms)	Activ.	Consulta	T(ms)
Coord.	1	3520	Coord.	2	3840	Coord.	3	4120
Part.	1	2750	Part.	1	2300	Part.	1	2800
Part.	2	16370	Part.	2	2910	Part.	2	2960
Part.	3	3250	Part.	3	2360	Part.	3	3410

Este es el caso de la ejecución de transacciones concurrentes en la base de datos distribuida, con replicación de datos, cuando ocurre una falla en una de las máquinas involucradas en una transacción global.
El Coordinador de la consulta 2 en Y, al no recibir respuesta del participante en X, decide abortarla e informa al resto esta decisión. Las otras dos consultas se ejecutaron con normalidad.
El tiempo de duración del participante donde se produjo la caída ascienden a 15000 milisegundos debido a que este tiempo es el definido para las caídas durante la simulación.

1.3: Falla simulada: Caída de X antes de que el Participante de la consulta 2 haya recibido el mensaje de "preparar" del Coordinador. Y caída de Y antes de que el Participante de la consulta 3 haya recibido el mensaje de "preparar" de su Coordinador.

Tablas de tiempos de ejecución:

Máquina X			Máquina Y			Máquina Z		
Activ.	Consulta	T(ms)	Activ.	Consulta	T(ms)	Activ.	Consulta	T(ms)
Coord.	1	18700	Coord.	2	1810	Coord.	3	3570
Part.	1	18290	Part.	1	17410	Part.	1	1790
Part.	2	16920	Part.	2	16090	Part.	2	1530
Part.	3	2750	Part.	3	16910	Part.	3	2690

Es el caso cuando ocurren fallas en dos de las máquinas involucradas en transacciones globales.
El Coordinador de la consulta 2 en Y, al no recibir respuesta del participante en X, decide abortarla e informa al resto esta decisión. El Coordinador de la consulta 3 en Z, al no recibir respuesta del participante en Y, decide abortarla e informa al resto esta decisión. Como X se cayo antes de que el Coordinador de la consulta 1 pudiera avisarle algo a los participantes, esta transacción nunca se llego a ejecutar.
Recordar que el tiempo de duración de las actividades donde se produjo la caída superan los 15000 milisegundos debido a que este tiempo es el definido para las caídas de las máquinas durante la simulación.

1.4: Falla simulada: Caída de X luego de que el Participante de la consulta 2 haya enviado el mensaje de "reconocimiento" final al Coordinador.

Tablas de tiempos de ejecución:

Máquina X			Máquina Y			Máquina Z		
Activ.	Consulta	T(ms)	Activ.	Consulta	T(ms)	Activ.	Consulta	T(ms)
Coord.	1	3790	Coord.	2	4780	Coord.	3	4340
Part.	1	2740	Part.	1	2580	Part.	1	2800
Part.	2	3620	Part.	2	3240	Part.	2	3410
Part.	3	3790	Part.	3	3300	Part.	3	3350

Este es el caso de la ejecución de transacciones concurrentes en la base de datos distribuida, con replicación de datos, cuando ocurre una falla en una de las maquinas involucradas en transacciones globales. Como la caída se produjo al final de la ejecución del protocolo, no hay muchas diferencias en los tiempos obtenidos durante la ejecución sin fallas de las transacciones.

Caso 2:

El ambiente simulado es de 2 máquinas que se llamarán X e Y. En todas ellas existen réplicas de las tablas UNO y DOS. El Manejador se encuentra en una tercer máquina para no interferir en el funcionamiento de los que serán participantes de la base de datos distribuida.

1. Ejecución de una transacción sobre UNO coordinándose en X.
2. Ejecución de una transacción sobre DOS coordinándose en Y.

2.1: Falla simulada: Sin fallas

Tablas de tiempos de ejecución:

Máquina X			Máquina Y		
Activ.	Consulta	T(ms)	Activ.	Consulta	T(ms)
Coord.	1	3020	Coord.	2	3190
Part.	1	2360	Part.	1	2140
Part.	2	1810	Part.	2	1700

El que se presentó es el caso de la ejecución normal de transacciones concurrentes en la base de datos distribuida, con replicación de datos. Ambas transacciones involucradas cometen satisfactoriamente.

2.2: Falla simulada: Caída de X antes de que el Participante de la consulta 2 haya recibido el mensaje de “preparar” del Coordinador.

Tablas de tiempos de ejecución:

Máquina X			Máquina Y		
Activ.	Consulta	T(ms)	Activ.	Consulta	T(ms)
Coord.	1	17410	Coord.	2	3290
Part.	1	2640	Part.	1	2470
Part.	2	16210	Part.	2	2370

Este es el caso de la ejecución de transacciones concurrentes en la base de datos distribuida, con replicación de datos, cuando ocurre una falla en una de las máquinas involucradas en una transacción global.
El Coordinador de la consulta 2 en Y, al no recibir respuesta del participante en X, decide abortarla e informa al resto esta decisión. La consulta restante (la 1) se logró ejecutar con normalidad antes de que cayera la máquina.

2.3: Falla simulada: Caída de X luego de que el Participante de la consulta 2 haya enviado el mensaje de "reconocimiento" final al Coordinador.

Tablas de tiempos de ejecución:

Máquina X			Máquina Y		
Activ.	Consulta	T(ms)	Activ.	Consulta	T(ms)
Coord.	1	2860	Coord.	2	3070
Part.	1	2040	Part.	1	2200
Part.	2	1430	Part.	2	1600

Ocurre la caída de una de las maquinas involucradas en transacciones globales al final de la ejecución del protocolo, por lo tanto no hay muchas diferencias en los tiempos obtenidos durante la ejecución sin fallas de las transacciones.

5. CONCLUSIONES Y LINEAS DE TRABAJO ACTUALES

La importancia del estudio de los sistemas distribuidos y particularmente en aplicaciones de tiempo real hace crecer la investigación experimental asociada con ellos. En este trabajo se expone una herramienta de simulación que ha demostrado ser particularmente útil para el estudio y predicción de comportamiento de SDTR en ambientes con arquitectura NOW (redes de workstations o PCs).
Si bien hasta el momento sólo se ha experimentado con un número limitado de sitios en el ambiente NOW y con casos específicamente orientados a BDD, la línea actual está centrada en repetir y extender las experiencias sobre una red homogénea de 8 procesadores existente en el LIDI y sobre una red heterogénea de 16 a 32 procesadores que también es utilizable desde el Laboratorio. Al mismo tiempo se extenderán el alcance de las fallas para estudiar simultáneamente migración y replicación de datos con migración y replicación de procesos.

6. BIBLIOGRAFIA

[BELL92] “Distributed Database Systems”. Bell, David; Grimson, Jane. Addison Wesley. 1992.

[COU95] “Distributed Systems. Concepts and Design”. George Coulouris, Jean Dollimore, Tim Kindberg. Addison Wesley. 1995.

[GOM97] “Software Design Methods for Concurrent and Real-Time Systems”. Hassan Gomaa. Addison Wesley. 1997.

[Hat88] “Strategies for Real-Time System Specification”. Hatley D., Pirbhai I. Dorset House, 1988.

[IEEE] Colección de "IEEE Transactions on Parallel and Distributed Systems", IEEE.

[JAVA00] “Java 1.2 Al descubierto”. Jaworski, Jamie. Prentice Hall, 2000.

[JAVAW1] “JavaTM 2 Platform, Standard Edition, v 1.3 - API Specification”. http://java.sun.com/j2se/1.3/docs/api/index.html

[JAVAW2] “The Java Tutorial - What is a Socket”. http://java.sun.com/docs/books/tutorial/networking/sockets/definition.htm

[MIA01] “Simulador para la evaluación de tiempos de respuesta en transacciones distribuidas y para el estudio de recuperación de errores”, Miaton Ivana, Tesina de Licenciatura en Informática, Universidad Nacional de La Plata, Argentina, 2001.

[Shu92] “Software specification and design for real-time systems”. Shumate K. Wiley 1992.

[TCPW1] “Protocolos TCP/IP”. Juan Salvador Miravet Bonet. http://www4.uji.es/~al019803/Tcpip.htm

[VAL99] Principles of Distributed Database Systems. M Tamer Ozsu; Patrick Valduriez. Prentice Hall, 1999.

[ZAN97] Análisis de Replicación en Bases de Datos Distribuidas, Zanconi, M, Tesis de Magister en Ciencias de la Computación, Universidad Nacional del Sur, Bahia Blanca, Argentina 1996.