Metodología de Sincronización de Relojes para Instrumentación

Fernando L. Romero, Fernando G. Tinetti[1]
Instituto de Investigación en Informática LIDI (III-LIDI)
Facultad de Informática – UNLP
fromero@lidi.info.unlp.edu.ar, fernando@info.unlp.edu.ar

CONTEXTO

Esta línea de Investigación forma parte de dos de los Subproyectos dentro del Proyecto "Sistemas Distribuidos y Paralelos" acreditado por la UNLP y de proyectos específicos apoyados por CyTED, CICPBA, Agencia Nacional de Promoción Científica y Tecnológica e IBM.

RESUMEN

En el presente trabajo se expone la línea de investigación sobre sincronización de relojes en ambientes distribuidos. Como parte de este trabajo se implementaron herramientas que permiten realizar evaluación de rendimiento a través de la instrumentación de código. Se desarrollaron estas herramientas en virtud de las falencias detectadas en las existentes. Si bien los experimentos se llevaron a cabo en ambientes de clusters, se estima en el futuro extender la herramienta a otros ambientes distribuidos utilizados para cómputo paralelo. Asimismo, las máquinas de dichos clusters son mono procesadores, pero se planifica investigar el problema aplicado a multiprocesadores y múltiples núcleos.

Keywords: *Sincronización de Procesos, Relojes Distribuidos, Rendimiento e Instrumentación, Sistemas Paralelos y Distribuidos, Paralelismo en Clusters e Intercluster, Sincronización Interna y Externa .*

1. INTRODUCCION

Los sistemas de cómputo y de comunicación que conforman un sistema distribuido disponen de relojes. En caso de querer realizar mediciones de eventos que se produzcan en diferentes nodos de dicho sistema, se hace necesaria la sincronización de estos relojes. En el caso de medir tiempos de comunicaciones, los requerimientos de exactitud con que se realizan esta sincronización y las mediciones han ido creciendo desde el segundo, milisegundo [6] [7], hasta llegar a exigir microsegundos [10] [9] [8]. Los tópicos que se estudian del problema comprenden:

- **Resolución**
- **Precisión**
- Tolerancia a fallas
- **Sobrecarga del sistema**
- **Exactitud**
- Fiabilidad
- Interoperabilidad
- Seguridad

La tarea de sincronizar relojes implica un compromiso entre estas variables, lo cual hace que por optimizar un aspecto, se degrada otro. Por ejemplo, si se incluye seguridad habría que encriptar los paquetes de comunicación de referencias horarias, lo cual degrada el aspecto de la sobrecarga, tanto de comunicaciones como el procesamiento de los mismos. Así es que en la lista anterior se resaltan

[1] Investigador Adjunto, Comisión de Investigaciones Científicas de la Provincia de Buenos Aires

en negrita los aspectos que se privilegian en este trabajo, esenciales al problema a resolver, la instrumentación. Por ello, se plantea la sincronización de relojes [3] [7] [6] pero con restricciones en los aspectos a resolver, de tal manera que cumplan los siguientes requerimientos:

- Pueda usarse como herramienta de instrumentación para programas paralelos
- Uso inicial en cluster de PC's, y posibilidad de ser extendido a clusters en general y a plataformas distribuidas aún más generales
- Sea de alta resolución, es decir, para medir tiempos cortos, del orden de microsegundos.
- No altere el funcionamiento de la aplicación a probar, o que la alteración sea mínima y conocida por la aplicación.
- Utilice en forma predecible la red de interconexión, determinando desde la aplicación los intervalos de tiempo en los cuales se usará la red, y así *desacoplar* el uso de la red entre tiempos para sincronización y tiempos para ejecución de programas paralelos.
- Se conozca el tipo (o al menos magnitud) de error con que se sincroniza.
- Sin incluir hardware adicional al del sistema. Para comunicaciones se usará la red entre computadoras que ya existe, y como sistema de medición el de cada computadora.

Como producto se obtuvo la biblioteca *st*, que realiza las tareas de sincronización dentro de los requerimientos previstos. También se obtuvo la biblioteca *timings*, para registro de tiempos, la cual provee la resolución adecuada, con bajo uso de recursos, sin usar entrada/salida ni llamadas al sistema, y con el menor uso posible de ciclos de CPU.

La exactitud fue uno de los tópicos priorizados, a fin de lograr cotas de error aceptables que hagan posible estimaciones de rendimiento en ambientes de red local de aplicaciones paralelas. Se debe tener en cuenta que en una red local Ethernet los tiempos de comunicación son del orden de decenas de microsegundos, por lo que determinaciones de tiempo transcurrido entre la partida y el arribo de un mensaje requieren sincronizaciones con diferencias de microsegundos en las máquinas que intercambian mensajes, para poder contar con una medida con un error aceptable.

En todas las circunstancias es deseable que el registro de tiempos implique la menor carga posible de procesamiento, como también conocer la magnitud del error que se comete en la medición. En el caso de mediciones de rendimiento, sería deseable que la tarea de sincronización se lleve a cabo fuera del tiempo en que se ejecuta el programa que se está monitorizando. Por otro lado, la cantidad de máquinas a sincronizar puede llevar a que el tiempo que se insume en la sincronización sea excesivo. Por ello se han llevado a cabo experimentos a fin de evaluar la escalabilidad de la herramienta en términos prácticos.

2. LINEAS DE INVESTIGACION Y DESARROLLO

La evaluación de rendimiento introduciendo instrumentación, exige que cada computadora cuente con un reloj físico, a partir del cuál se derivan los relojes lógicos que son los que se sincronizan [4]. De las limitaciones de estos relojes dependerá lo que se pueda lograr, por lo que se realizaron experimentos a fin de caracterizar estos relojes en cuanto a exactitud, estabilidad, confiabilidad, resolución y demás atributos.

Los relojes lógicos deben sincronizarse en una ***hora inicial*** a partir de la cual se empiezan a actualizar y en ***frecuencia*** de actualización de la misma. Las referencias son intercambiadas entre las máquinas a través de la red existente. Esto adiciona una demora que debe adicionarse a la referencia. Debido a la varianza en los tiempos de comunicaciones, es difícil determinar este valor más allá de cierta exactitud. Para la medición de este valor se puede recurrir a la estadística de los valores de ida-vuelta (*round trip time*) de un mensaje y asumir que solo serán validos los valores de referencia trasmitidos en un tiempo de transmisión igual a la ***moda*** de los tiempos de transmisión. Ya que la información es provista y centralizada en el servidor, al aumentar la cantidad de clientes, el servidor se transforma en un cuello de botella. De tal manera que la escalabilidad del modelo debió probarse.

3. RESULTADOS OBTENIDOS/ESPERADOS

A partir del estudio de los algoritmos básicos e implementaciones existentes [1] [2] [6], se realizaron un conjunto de experimentos a fin de poder compararlas y ver sus limitaciones [9] [10] [11]. Como conclusión de dichos experimentos se desarrolló una biblioteca *timings* para mediciones de tiempo a nivel local y una biblioteca *synchro*, que funciona en forma distribuida y permite sincronizar en hora y frecuencia los relojes de *timings*. Estas bibliotecas además proveen una caracterización del error, dada por sus diversos componentes:

- Error en la estimación del tiempo de comunicación entre máquinas de las referencias.
- Error debido a la precisión del reloj local de la biblioteca desarrollada (*timings*).
- Error debido a latencia propia del sistema operativo.

Como extensiones futuras, siempre es deseable la sincronización externa de los relojes [3]. Este paso está muy ligado también a la posibilidad de utilizar más de un cluster de computadoras para cómputo paralelo y en este contexto la sincronización de los relojes va más allá del análisis de rendimiento con el objetivo de optimizarlo.

Una de los experimentos fue medir la escalabilidad del modelo [10] [11], midiendo el tiempo en microsegundos que lleva sincronizar los relojes de 2 a 16 computadoras. Los resultados resumidos se muestran en la siguiente tabla:

	2	4	6	8	10	12	14	16
Mínimo	37972	39902	40819	38596	41910	37522	38525	41774
Máximo	43362	46933	52908	48233	59397	53436	56029	52791
Promedio	40475	42444	45314	43585	47353	45433	47645	48312

Tabla I: Tiempos de ejecución para 20 corridas de Synchro en 2 a 16 máquinas.

En forma gráfica se puede apreciar la evolución de estos tiempos en la Fig. I, donde, además, se muestran los valores intermedios que se omitieron en la tabla I.

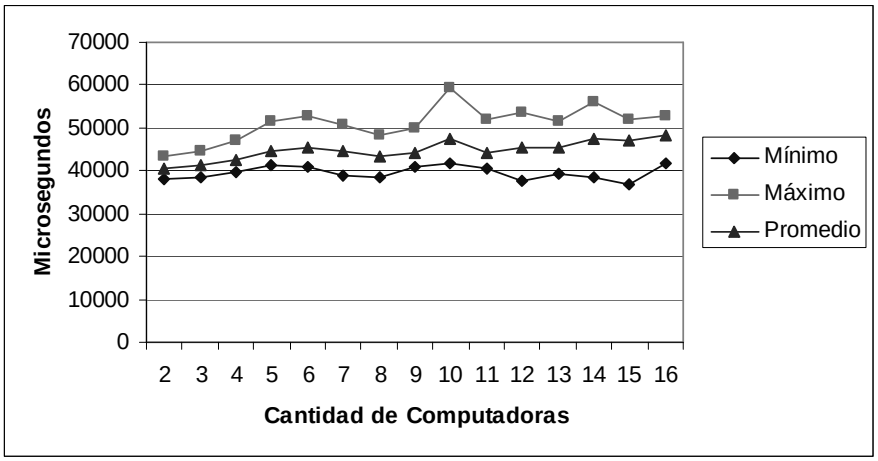

Figura I: Tiempos de ejecución para 20 corridas de Synchro en 2 a 16 máquinas.

Estos resultados permiten concluir que la escalabilidad de este algoritmo, a pesar de basarse en un modelo cliente-servidor, no representa mayores incovenientes en los tamaños típicos de clusters. La sincronización se lleva a cabo entre el servidor y cada cliente, en forma secuencial. Como se desprende de restar al tiempo de sincronizar 16 el de sincronizar 2, y dividir dicho tiempo por 14 máquinas, corresponderían 560 microsegundos por máquina agregada a la sincronización. Los tiempos máximos medidos son menores a 60 milisegundos en total. Teniendo en cuenta la cantidad de computadoras de un cluster, son valores despreciables, sobre todo si se piensa que son tiempos

previos al proceso de medición, y absolutamente aislados del mismo.
La uniformidad y linealidad de los tiempos, permite estimar extrapolaciones. Sería posible estimar en 560 x 100 = 56000, o sea, 56 milisegundos el tiempo agregado a los 40 milisegundos iniciales para sincronizar un cluster de 100 computadoras, llevando el total del tiempo de sincronización a menos de una décima de segundo para un cluster de 100 computadoras. En experimentos posteriores, se comprobó que los aproximadamente 40 milisegundos iniciales se deben a falta de sincronización entre clientes y el servidor. Al sincronizarse los clientes con un *delay*, desaparece dicho tiempo inicial, ya que solicitan sincronizarse recién cuando el servidor ya terminó de inicializarse.
Con respecto a sobrecarga de tiempo de medición durante la ejecución de un programa paralelo, cada consulta de tiempo implica un tiempo de entre 230 y 340 ciclos de reloj de CPU [7], con lo cual, en una computadora con velocidad de reloj 2Ghz, estaríamos en el orden de 1 a 2 décimas de microsegundo por cada medición, contra 615 ciclos para gettimeofday(), que es una llamada al SO, opcional al método utilizado por la biblioteca *timings*,.
Otro experimento que se llevó a cabo, es la utilización de una única referencia, a cargo del servidor, para determinar la constante de ***sincronización de frecuencia*** MHz. La tabla II muestra los valores que se obtienen al sincronizar relojes con cálculo de MHz totalmente independientes (con los relojes locales) y con cálculo de MHz con referencia única.

Cant. Máq.	**Ref. única**		**Ref. local**	
	Mín.	Máx.	Mín.	Máx.
	Us	us	Us	us
2	-1	3	42	103
4	-3	3	55	98
8	-3	4	56	325
16	-4	3	43	1109

Tabla II: Errores de Sincronización por cálculo de MHz.

En el cálculo de MHz con referencia única (mostrado en la Tabla II) una sola computadora proporciona las referencias de tiempo para el cálculo de MHz. El experimento se llevó a cabo sincronizando desde 2 hasta 17 máquinas, y luego se midió la diferencia 1 segundo después. En todos los casos, son valores mínimos y máximos luego de 50 corridas.
Se pueden comparar estos valores con los obtenidos en NTP. La tabla III muestra dicha comparación. Los datos evitan cualquier comentario sobre la superioridad de la biblioteca *st* respecto de NTP [6] [7] [8] para sincronizar la hora en un cluster, sobre todo, teniendo en cuenta que la experiencia se realizó sobre el mismo hardware:

	NTP		Biblioteca *st*	
	Máximo	Estimado	Máximo	Estimado
Min	239673	31	32	1
Max	989368	2434	33	4

Tabla III: Comparación de error en sincronización de NTP contra la Bilblioteca *st*.

También se realizaron pruebas tendientes a verificar la estabilidad de frecuencia [4] de los relojes de cuarzo en el tiempo similares a las reportadas en [5]. El experimento requeriría de una referencia de tiempo perfecta o al menos con la menor deriva de frecuencia posible, tal como un reloj atómico o la hora UTC. Debido a que no se cuenta con estos elementos, se toma como referencia el reloj de cuarzo del *timer* de una PC, y con el se miden las variaciones del reloj TSC (*Time Stamp Counter*). Si bien no se puede medir el error en términos absolutos, se puede determinar cuánto varía la diferencia de deriva. Dado que en la línea de investigación que se sigue no se requiere de referencias externas, sino solo que las referencias estén ajustadas a la escala de tiempos que

determine el servidor, es válido medir esta variación de frecuencia y pensar que el error será proporcional a dicha diferencia. Los resultados obtenidos son similares a los de [5], de menos de 10 ppm (partes por millón) en un período de varias horas.
Para la definición de futuras extensiones, se tienen planificados algunos experimentos para estimar:

- La mejora proporcionada por la comunicación de la referencia de tiempo utilizando comunicación *broadcast*.
- Más exhaustivamente la estabilidad de los relojes de las máquinas y su deriva, a fin de determinar la relación del intervalo entre sincronizaciones con el error de sincronización esperado.
- La utilidad de un servidor de hora con sistema operativo de tiempo real para disminuir la latencia impuesta por el SO al menos del lado del servidor.

4. FORMACION DE RECURSOS HUMANOS

En esta línea de I/D existe cooperación a nivel nacional e internacional. Se ha completado una tesis de maestría y está abierta la posibilidad para varias Tesinas de Grado de Licenciatura.

5. BIBLIOGRAFIA

[1] G. Coulouris, Dollimore J., Kindberg T., Sistemas Distribuidos. Conceptos y Diseño, 3ª edición. Pearson Educación, 2001, ISBN: 8478290494.
[2] F. Cristian. "Probabilistic Clock Synchronization", *Distributed Computing*, 3: 146–158, 1989.
[3] Cristian F., Fetzer C., "The Time Asynchronous Distributed System Model", IEEE Transactions on Parallel Systems, June 1999, pp. 603618.
[4] X. Luo, "TSC-I2: A Lightweight Implementation for Precision-Augmented Timekeeping", reporte técnico de University of Illinois at Chicago, http://tsc-xluo.sourceforge. net/TSC-I2_Tech_Report.pdf
[5] D. L. Mills, "Measured performance of the Network Time Protocol in the Internet System". ACM Computer Communication Review 20, Jan. 1990. pp. 6575.
[6] D. L.Mills, "A Brief History of NTP Time: Confessions of an Internet Timekeeper". ACM Computer Communications Review 33, 2 (April 2003), pp 922.
[7] Mills D.L., "Network Time Protocol (Version 3) specification, implementation and analysis", DARPA Networking Group Report RFC1305, University od Delaware, March 1992.
[8] Mills D. L., Kamp P.H., "The Nanokernel", Proc. Precision Time and Time Interval (PTTI) Applications and Planning Meeting (Reston VA, November 2000).
[9] F. L. Romero, W. Aróztegui, F. G. Tinetti, "Sincronización de Relojes en Ambientes Distribuidos", XII Congreso Argentino de Ciencias de la Computación (XII CACIC) Octubre 2006.
[10] F. L. Romero, Armando E. De Giusti, Fernando G. Tinetti. "Sincronización de Relojes para Evaluación de Rendimiento: Experiencias en un Cluster Utilizado para Cómputo Numérico". XIV Congreso Argentino de Ciencias de la Computación (XIV CACIC) Octubre 2008
[11] Fernando G. Tinetti, Fernando L. Romero, Armando E. De Giusti "Clock Synchronization in Clusters for Performance Evaluation: Numeric/Scientific Computing". World Congress on Computer Science and Information Engineering (CSIE 2009), Abril 2009.