Sincronización de Relojes en Ambientes Distribuidos

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 e IBM.

RESUMEN

Esta línea de investigación se orienta a resolver el problema de sincronización de tiempo en ambientes distribuidos. El objetivo inicial de la sincronización de relojes es la estimación de rendimiento a partir de experimentos con la mínima instrumentación (y su consecuente interferencia) posible. El algoritmo básico sigue las estrategias clásicas de sincronización de tiempo en ambientes distribuidos. Sin embargo, se considera necesario adaptar estas estrategias al entorno de un cluster o, al menos, de una red de interconexión sobre la que se tiene acceso exclusivo (o *controlado*) para todas las comunicaciones entre las computadoras que se sincronizan. Este ambiente es específicamente el de los entornos de cómputo paralelo en clusters.
Keywords: *Sincronización de Procesos, Relojes Distribuidos, Rendimiento e Instrumentación, Sistemas Paralelos y Distribuidos, Paralelismo en Clusters e Intercluster, Sincronización Interna y Externa .*

1. INTRODUCCION

Siempre fue necesario disponer de una referencia de tiempo en los sistemas de cómputo [13] [15]. Dicha referencia es esencial para resolver problemas tales como el ordenamiento de eventos (ej: envío y recepción de correo electrónico, eventos dentro de las transacciones, inicio de procesos en tiempo real, etc.). También cobra importancia la medición de tiempos en la optimización del rendimiento tanto en sistemas monoprocesador como en sistemas paralelos y distribuidos [3] [6] [13]. En todos los casos, las aplicaciones con fuertes requerimientos de cómputo o procesamiento son las que también requieren la optimización para el máximo aprovechamiento del hardware disponible. La relación es bastante directa: a partir de la monitorización de los tiempos de ejecución se pueden analizar los problemas de rendimiento e intentar solucionarlos [8].
Aún en el caso de las mediciones en una misma computadora (usualmente con un único procesador), es deseable que el registro de tiempos no influya en el tiempo de ejecución de la misma. En todos los casos, se necesitan resoluciones de reloj acordes a los tiempos que se deben medir en las aplicaciones. Usualmente, los métodos provistos por el sistema operativo no son apropiados [14] como también los métodos y/o herramientas provistas por los lenguajes que dependen del sistema operativo.
En el caso de procesamiento distribuido, con un programa que ejecuta procesos en diferentes computadoras o en los que el tiempo de las comunicaciones es importante, la tarea de medir intervalos de tiempo implica sincronizar los relojes de las diferentes computadoras que se utilizan [4] [11]. Sería deseable que esta tarea de sincronización se lleve a cabo fuera del tiempo en que se

[1] Investigador Asistente CICPBA

ejecute el programa que se está monitorizando, y conociendo el tipo (o al menos magnitud) de error con que se sincroniza.
Es deseable que dicha sincronización se lleve a cabo sin la necesidad de incluir hardware adicional al del sistema, con lo que las comunicaciones deberán utilizar la red de interconexión entre computadoras existente y el sistema de medición existente en cada sistema de cómputo. Se desea contar con una herramienta de instrumentación para programas paralelos que:

Pueda ser usada inicialmente en un cluster de PC's, con la posibilidad de ser extendido a clusters en general y luego en plataformas distribuidas aún más generales.

Sea de alta resolución, es decir que se pueda utilizar para medir tiempos cortos, del orden de microsegundos.

Que no altere el funcionamiento de la aplicación bajo prueba, o que la alteración sea mínima y conocida por la aplicación.

Utilice en forma predecible la red de interconexión. Más específicamente, se puedan determinar, desde la aplicación, los intervalos de tiempo en los cuales se utilizará la red. De esta forma, se puede *desacoplar* el uso de la red de interconexión, ya que habrá intervalos de tiempo usados para la sincronización e intervalos de tiempo utilizados para la ejecución de programas paralelos.

2. LINEAS DE INVESTIGACION Y DESARROLLO

Inicialmente, se **estudian tanto los algoritmos básicos como las implementaciones existentes** [1] [2] [10]. Como requisito previo, se establece que cada computadora cuente con un oscilador físico de frecuencia relativamente constante. A partir de este oscilador físico se derivan los relojes lógicos que son los que se sincronizan [9]. En todos los casos, lo que se tiende a resolver son las diferencias de [17]:

1. Referencia fija en el tiempo a partir de la cual se contabiliza el tiempo en cada computadora. Aunque normalmente es constante, es relativamente difícil establecer con precisión su valor.
2. Frecuencia entre los relojes de las computadoras que se sincronizan.

Utilizando los fundamentos de los algoritmos básicos y las implementaciones existentes, se adaptan para tener en cuenta los requisitos mencionados y **se implementa un algoritmo específico para la sincronización de relojes en un cluster**. Este algoritmo debe ser caracterizado en términos de los dos parámetros mencionados: referencia fija en el tiempo y frecuencias de los relojes físicos que intervienen.
Una vez realizada la sincronización mínima entre las computadoras se debe **investigar el comportamiento de la misma en términos de escalabilidad**. Usualmente la sincronización se da entre dos máquinas y en el caso particular de NTP (Network Time Protocol) se lleva a cabo con el modelo cliente/servidor. Es claro que cualquier tipo de centralización (en el servidor, por ejemplo) tiene sus inconvenientes de escalabilidad y al menos debería ser posible su cuantificación. En este contexto específico es muy interesante la posibilidad de sincronización utilizando mensajes *broadcasts* con su consiguiente ahorro de comunicaciones punto a punto.
Como extensiones futuras, siempre es deseable la **sincronización externa de los relojes** [7]. 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.

3. RESULTADOS OBTENIDOS/ESPERADOS

Se desarrolló una biblioteca de medición de tiempos de cómputo dentro de en un mismo sistema operativo (usualmente monoprocesador) para Linux en PCs (procesadores Intel y compatibles) en las cuales se puede hacer referencia al contador de ciclos del oscilador. En este contexto, se han llevado a cabo mediciones de los ciclos de CPU invertidos en una llamada al sistema como gettimeofday y la utilizada por la alternativa presentada (con RDTSC) en diferentes computadoras.

La Tabla 1 muestra los valores obtenidos en ciclos de CPU en las computadoras con las características que se dan en la Tabla 2. También a partir de los datos de la Tabla 1 (ciclos de CPU, específicamente) y los de la Tabla 2 (MHz de reloj, específicamente) se comprueba que la precisión es acorde a los requerimientos, del orden de los microsegundos.

Tabla 1: Ciclos de CPU de gettimeofday y RDTSC.

PC	Gettimeofday (ciclos)	Rut. RDTSC (ciclos)	Precisión RDTSC (μs)
P42400	7876	2828	≅1.18

Tabla 2: Detalles de la PC de la Tabla 1.

PC	CPU	Frecuencia (MHz)	Sistema Operativo
P42400	Intel Pentium 4	2400	Linux 2.4.18-14

Se debe notar que la sobrecarga (tiempo *extra* de ejecución que no pertenece a la aplicación de usuario) de esta rutina de medición desarrollada es igual a la precisión de la Tabla 1, dado que no hay cambios de contexto al incluir la biblioteca en el binario de la aplicación. En el experimento se provocó el desalojo de gettimeofday y rdtsc de memoria caché tanto de instrucciones como de datos, que es la condición real en la que serán usados en instrumentación.
Además, se desarrolló una biblioteca de una cantidad reducida de funciones para instrumentación de programas paralelos que se ejecuten en clusters de PCs. Se siguieron los lineamientos dados antes: resolución del orden de los microsegundos, la mínima sobrecarga de procesamiento y el desacople de la red de interconexión. La tabla 3 muestra los resultados usando un método de única referencia de reloj para el cálculo de los MHz de todas las computadoras y usando la referencia local (el método *estándar*).

Tabla 3: Diferencias en microsegundos (Valores Absolutos).

Cantidad	Ref. única		Ref. local	
	Mín.	Máx.	Mín.	Máx.
2	1	3	42	103
3	3	3	56	116
4	3	3	55	98
5	2	3	55	114
6	2	3	57	244
7	3	3	56	1533
8	3	4	56	325
9	3	3	56	210
10	3	4	4	1308
11	4	4	41	1395
12	4	5	58	1802
13	4	5	57	1277
14	3	3	43	3808
15	3	4	43	1853
16	4	3	43	1109
17	14	6	50	2128

El experimento se llevó a cabo sincronizando de 2 hasta 17 máquinas, y luego midiendo la diferencia 1 segundo después. En todos los casos son valores mínimos y máximos luego de varias corridas. La forma de calcular la frecuencia de reloj de todas las computadoras con una única computadora (normalmente un “servidor”) debe tener en cuenta algún tipo de interconexión entre ellas con el consiguiente tiempo de las comunicaciones de sincronización. Básicamente, se

determina una constante de corrección de la frecuencia a través de mensajes con la máquina de referencia: se inicia una medición de tiempo lo suficientemente largo como para que sea despreciable frente al tiempo de comunicaciones, y se calcula en función de la relación entre los ciclos de reloj. Queda pendiente la posibilidad de extensión a otros sistemas operativos y/o plataformas de hardware.

Por otro lado, se analizaron y cuantificaron características de NTP (Network Time Protocol, usado de manera estándar para la sincronización de relojes distribuidos) referidas a la precisión del reloj sincronizado. Hay dos precisiones involucradas, la del reloj local (gettimeofday) que en los casos vistos es de aproximadamente 3.3μs. y la de NTP. En NTP la precisión es determinada en forma automática y medida como potencia de 2. Para la máquina bajo experimento:

-17, o sea 2e-17= 7,6μs.

Este análisis es básicamente experimental [11] [12], dado que el funcionamiento de NTP se define por un lado paramétricamente y por el otro con el comportamiento o rendimiento de la red de interconexión. En la tabla 4 se pueden observar los valores obtenidos en una máquina tratando de sincronizar con un servidor en red local. Se puede observar que alcanzar la sincronización llevó un tiempo de 6 minutos y que a partir de allí no mejora demasiado la sincronización. El experimento llevó varias horas, los resultados están recortados en la tabla a partir de no observarse mejoría.

Tabla 4: Experimentos con NTP.

Hora	Offset(μseg)	Dist. Sync. (mseg)	synch?	Stratum(ref:12)
15:46:14	0,000018	0,00375	Not	16
15:47:00	0,000019	0,00444		
15:49:00	0,000016	0,45090		
15:50:00	0,000016	0,45180		
15:51:00	0,000018	0,45271		
15:52:00	-0,000369	0,01141	Yes	12
15:54:00	-0,000442	0,01126		
15:55:00	-0,000446	0,01120		
15:56:00	-0,000451	0,01111		
15:58:00	-0,000457	0,01097		
16:00:00	-0,000455	0,01178		

Una vez que se determina el valor de offset, el límite del error aumenta en una tasa fija debido al error de frecuencia máximo del oscilador del reloj. A este límite se lo denomina distancia de la sincronización a la fuente (stratum 0), y es medido estadísticamente por NTP. Offset es la diferencia de hora respecto a dicha fuente. El intervalo de corrección se define como dos veces este límite con el punto medio igual al valor de offset. Este es el intervalo dentro del cual se puede decir que se está sincronizando. En el experimento se efectúa sincronización interna, sin utilizar un servidor de hora externo y sincronizando una máquina contra la otra, se obtiene una sincronización mayor a 400μs (Offset), con un error en dicha medida del orden de los 11ms (Dist. Sync.).

Relativo al timed de Linux (que implementa el algoritmo de Berkeley [1]), se realizaron pruebas, pero la herramienta para medir diferencias de tiempos una vez que las máquinas se sincronizan, tiene una resolución de 1ms., reportando en el mejor de los casos diferencias de entre 0 y 1 ms.

Queda pendiente un análisis de los sistemas que utilizan hardware especializado, como para tener un marco de referencia más amplio [5]. En todos estos casos, la idea final es contar con un conjunto de programas del estilo de los *benchmarks* para que provean automáticamente la caracterización del sistema de sincronización elegido/utilizado. También se espera extender la biblioteca para su uso en Internet y múltiples clusters para cómputo paralelo [16]. Quizás en este punto se deban redefinir algunas características de la herramienta o biblioteca, tal como la capa de transporte de los mensajes con los cuales se implementa la sincronización.

4. FORMACION DE RECURSOS HUMANOS

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

5. BIBLIOGRAFIA

[1] Coulouris G., 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., C. Fetzer. "The Timed Asynchronous Distributed System Model", IEEE Transactions on Parallel and Distributed systems, June 1999, pp. 603618.
[4] Cristian F., Fetzer C., "The Time Asynchronous Distributed System Model", IEEE Transactions on Parallel Systems, June 1999, pp. 603618.
[5] Elson K. J., Romer K., "Wireless Sensor Networks: A New Regime for Time Synchronization in Distributed Systems", Proceedings of the First Wirkshop on Hot Topics In Networks (HotNets1), Princeton, New Jersey, October 2002.
[6] Elson K. J., Girod L., Estrin D., "FineGrained Network Time Synchronization using Reference Broadcasts", Proceedings of fifth symposium on Operating System Design and Implementation. December 2002.
[7] Fetzer C., Christian F., "Integrating External and Internal Clock Synchronization", June 1996.
[8] Grove D. A.. "Performance Modelling of messagepassing parallel programs", May 2003.
[9] Kim K. H., Im C., Athreya P, "Realization of a Distributed OS Component for Internal Clock Synchronization in a LAN Environment", Proc. ISORC 2002, IEEE 5th Int'l Symp on Objetoriented Realtime distributed Computing, Washington, D.C., April 2002, pp. 263270.
[10] Mills D. L. "Measured performance of the Network Time Protocol in the Internet System". ACM Computer Communication Review 20, Jan. 1990. pp. 6575.
[11] Mills D. L., "A Brief History of NTP Time: Confessions of an Internet Timekeeper". ACM Computer Communications Review 33, 2 (April 2003), pp 922.
[12] Mills, D.L. "Unix kernel modifications for precision time synchronization". Electrical Engineering Department Report 94101, University of Delaware, October 1994.
[13] Mills, D.L, Kamp P.H., "The Nanokernel", Proc. Precision Time and Time Interval (PTTI) Applications and Planning Meeting (Reston VA, November 2000).
[14] Rubini A., Corbet J., Linux Device Drivers, 2nd Edition, ISBN 0596000081, June 2001.
[15] Work P., Nguyen K., "Measure Code Sections Using The Enhanced Timer", http://www.intel.com.ar, October 2005.
[16] Zhao Y., Zhou W., Huang J, Yu S., "Self Adaptive Clock Synchronization for Computational Grid", Journal of Computer Science and Technology, 2003, Volume: 18, Issue: 4, pp. 434 – 441.
[17] Fernando L. Romero, Walter Aróztegui, Fernando G. Tinetti, "Sincronización de Relojes en Ambientes Distribuidos", XII Congreso Argentino de Ciencias de la Computación (XII CACIC) Octubre 2006.