Problemas de la Sincronización de Relojes en Clusters

Fernando L. Romero, Fernando G. Tinetti*

III-LIDI, Facultad de Informática, UNLP
50 y 115, 1900, La Plata, Argentina
fromero@lidi.info.unlp.edu.ar, fernando@info.unlp.edu.ar

Resumen

Se presenta un trabajo orientado a resolver algunos aspectos del problema de sincronización de relojes en ambientes distribuidos. El objetivo inicial está dirigido a la estimación de rendimiento en estos ambientes, donde es necesario algún tipo de sincronización de relojes de las computadoras que se utilizan. El algoritmo básico que se utiliza es una modificación del algoritmo de Cristian, considerando necesario adaptarlo al entorno de un cluster o, al menos, de una red de interconexión sobre la que se tiene información y acceso relativamente exclusivo o previsible para todas las comunicaciones entre las computadoras que se sincronizan. Una de las ideas subyacentes es la de aislar la sincronización de la ejecución de la aplicación cuyo rendimiento se quiere evaluar. La finalidad de este requerimiento es evitar una intervención de la instrumentación sobre los valores a medir, o al menos saber qué error tendrán los mismos. El ambiente en el cual se realizan los experimentos es inicialmente el de los entornos de cómputo paralelo en clusters. Se ensayan diferentes métodos para la sincronización en valor inicial y frecuencia de los relojes.

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

Abstract

This paper is oriented to solve some of the aspects of the clock synchronization problem in distributed environments. Its main objective is focused on the performance estimate in these environments, in which it is necessary to count with some kind of clock synchronization of the computers used. The basic algorithm used is a modification of the Christian's algorithm, which must be adapted to a cluster's environment or, at least, to an interconnection network about which some information is available and to which there exist a relatively exclusive or predictable access for all the communications among the synchronized computers. One of the underlying ideas is that of isolating the application running synchronization, whose performance is to be evaluated. The end sought with this requirement is to avoid the intervention of instrumentation over the values to be measured or, at least, to know which errors they might present. The environment in which experiments are carried out is initially that of cluster parallel computing environments. Different methods are tested for the synchronization in initial values and clock frequency.

Keywords: *Process Synchronization, Distributed Clocks, Performance and Instrumentation, Parallel and Distributed Systems, Parallelism in Clusters and Interclusters, Internal and External Synchronization.*

* Investigador Asistente CICPBA

1 INTRODUCCION

Las computadoras poseen uno o varios relojes que proporcionan una referencia de tiempo [16] [22]. 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.) como para tareas propias del sistema, tales como planificación de procesos. Estos relojes permiten la medición de intervalos de tiempo, necesarios para la optimización del rendimiento tanto en sistemas monoprocesador como en sistemas paralelos y distribuidos [15] [4] [10]. En todos los casos, las aplicaciones con fuertes requerimientos de cómputo, procesamiento y comunicaciones son las que también requieren la optimización para el máximo aprovechamiento del hardware disponible. A partir de la monitorización de los tiempos de ejecución se pueden analizar los problemas de rendimiento e intentar solucionarlos [6].

Al realizar mediciones de tiempo de ejecución de procesos y de comunicaciones es deseable que el registro de tiempos no influya en el tiempo de ejecución de los mismos. Debido al incremento de las velocidades de los procesadores, la disminución de las latencias en las comunicaciones y el aumento del ancho de banda en las mismas, dado por la evolución del hardware, el requerimento de resolución en las medidas de tiempo fue creciendo, estando en la actualidad en el orden de los microsegundos. Los métodos provistos por el sistema operativo no son apropiados [17] debido no solo a problemas de resolución sino también por el nivel de interferencia (Ej: llamadas al SO). Los métodos y/o herramientas provistas por los lenguajes heredan este problema ya que dependen del sistema operativo. La solución es una mejor uso de las nuevas características del hardware. Ejemplo es el uso de la instrucción RDTSC (ReaD Time Stamp Counter) [16] [22].

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 [9]. Sería deseable que esta tarea de sincronización se lleve a cabo fuera del tiempo en que se ejecute el programa que se está monitorizando, y conociendo el tipo (o al menos magnitud) de error con que se sincroniza. Es requerimiento de este trabajo 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. Asimismo, como sistema de medición, aprovechar el hardware de cada sistema de cómputo, sin agregados ni cambios.

En principio, esta herramienta de instrumentación sería utilizada para evaluar programas paralelos, cumpliendo 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, que 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 TRABAJOS PREVIOS RELACIONADOS

Se estudiaron tanto los algoritmos básicos como las implementaciones existentes. Como requisito previo, se establece que cada computadora cuente con un reloj físico de frecuencia más o menos constante. En general, estos relojes utilizan como estabilizador del oscilador un cristal de cuarzo. Presentan valores de variabilidad de frecuencia que van desde 10^{-4} a 10^{-6} [18]. A partir de este reloj físico se derivan los relojes lógicos que son los que se sincronizan [5] [8] [13]. En todos los casos, lo que se tiende a resolver son las diferencias de [19] [20]:

1. Referencia fija en el pasado a partir de la cual se contabiliza el tiempo en cada computadora.
2. Frecuencia entre los relojes de las computadoras que se sincronizan.

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. La propuesta inicial de Cristian [3], de contar con un servidor que proporcionara una hora ajustada a un determinado estándar (sincronización externa) y clientes sincronizando con este servidor, sirvió de base para el desarrollo de protocolos más complejos en los que se consideraron otros aspectos del problema. Podemos citar a NTP (Network Time Protocol) y el algoritmo de Berkeley como ejemplos, siendo este último un ejemplo de sincronización interna. En cuanto a ordenación de eventos sin un estándar en la hora, el algoritmo de Lamport es un ejemplo de otra orientación para solucionar el problema de sincronización. 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. Referido a análisis y experimentación de sistemas existentes, se han llevado a cabo pruebas con NTP y con el Algoritmo de Berkeley específicamente con su implementación en Linux, el timed (**time d**aemon). También se analiza a continuación el algoritmo de Lamport.

2.1 NTP: Network Time Protocol

Una de las primeras consecuencias de utilizar NTP está relacionada con la precisión del reloj obtenido, independientemente de la sincronización. En el caso de la instalación de NTP realizada en computadoras con Linux 2.4.18-14, sobre PCs con procesador Intel Pentium 4 de 2.4 GHz y 1 GB de RAM, la precisión del reloj de NTP es de 2e-17s = 7,6 μs. En estas mismas computadoras, la resolución del reloj sin NTP es de 1 μs, con lo que en cierta forma se pierde algo de precisión *a priori*. NTP corrige dos tipos de diferencias (*offsets*) para sincronizar dos o más computadoras:

- ***Offset* de hora:** Va realizando ajustes de acuerdo a que la diferencia respecto de la referencia sea grande o pequeña. Si la diferencia es mayor a 128 μs. Se da en un escalón (*stepping*) y si es menor a 128ms, en forma gradual (*slewing*). En todos los casos, si el reloj local esta adelantado, va haciendo ajustes graduales de tal manera de no producir discontinuidades en la hora (hora posterior < hora actual).
- ***Offset* de frecuencia:** a partir de analizar la primera derivada de la diferencia de horas, establece si la frecuencia es mayor o menor a la de la referencia, cambiando el valor local de frecuencia al correcto.

En la Tabla 1 se pueden observar algunos valores relacionados con la sincronización de una máquina con respecto a un servidor en la misma red local. Las computadoras son iguales, con las características ya mencionadas, y la red de interconexión es Ethernet de 100 Mb/s dedicada (no hay otro tráfico en la red) con *switch* de capa 2. Mientras no posee suficientes datos como para asegurar sincronización, NTP informa que no está sincronizada, y el valor de hora dependerá del reloj local. A partir de alcanzar una sincronización mínima a los 6 minutos de haber comenzado, NTP ya proporciona los valores corregidos. A partir de este primer instante de sincronización, los valores

relacionados mejoran considerablemente, aún después de varias horas de funcionamiento de NTP.

Tabla 1: Valores de Sincronización de NTP.

Hora	*Offset*(s)	*Dist. Sync.(s)*	*synch?*
15:46:14	0,000018	0,00375	Not
15:47:00	0,000019	0,00444	
15:48:00	0,000022	0,00534	
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
15:53:00	-0,000431	0,01134	
15:54:00	-0,000442	0,01126	
15:55:00	-0,000446	0,01120	
15:56:00	-0,000451	0,01111	
15:57:00	-0,000459	0,01105	
15:58:00	-0,000457	0,01097	
15:59:01	-0,000461	0,01187	
16:00:00	-0,000455	0,01178	

El valor que proporciona NTP para estimar la magnitud del error de sincronización en un instante dado es la distancia de sincronización (Dist.Sync. en la Tabla 1). Este parámetro se calcula en base a una estadística en los tiempos de comunicación con el servidor de referencia, la distancia de sincronización con que el servidor a su vez recibe los valores de referencia, y los valores registrados de *offset* de frecuencia. Las variaciones de frecuencia (o posible derivada segunda de la hora) se registran e incorporan al cálculo de la distancia de sincronización. El pseudocódigo del cálculo de dicho parámetro puede verse en el RFC-1305.

La dificultad de contar con la distancia de sincronización como única estimación de error, es que no se puede saber con certeza el error de sincronización sino una cota, que normalmente es del orden de los milisegundos. En la Tabla 1 se puede verificar que se sincroniza a valores de alrededor de 400 μs (*offset*) con un posible error acotado de poco más de 1100 μs, valores alejados de los requeridos en este trabajo. Con respecto a la sobrecarga en comunicaciones, se puede configurar la frecuencia a la que se interrogará al servidor. En caso de querer eliminar toda intrusión de NTP en las medidas de rendimiento que se deban realizar, sería necesario detener los procesos de NTP, con lo que los valores de error aumentarán, y en forma imprevisible, pues el reloj empezará a tomar como referencia el reloj local, con las correcciones hechas por NTP en el período en que estuvo funcionando. De otra manera, habría que admitir el "ruido" de los paquetes de sincronización, normalmente en sucesiones (o *ráfagas*) de 4 paquetes UDP, que también implican el costo de procesamiento por parte de los de NTP, con las posibles llamadas al sistema para ajustar los valores de la hora.

2.2 Algoritmo de Berkeley

A diferencia de NTP, el algoritmo de Berkeley tiene un proceso (o computadora) *Master* que se encarga de definir un tiempo sincronizado a partir del tiempo local de todos los demás procesos (o computadoras). De manera periódica, el Master

1. Requiere la hora local de cada uno de los demás procesos.
2. Calcula el promedio de los valores recibidos (descartando los que difieran demasiado).

3. Informa a los demás procesos el cambio que deben realizar para estar sincronizado.

En el cálculo de cada nuevo valor de tiempo a utilizar (con el promedio de los tiempos recibidos) no se tiene en cuenta el tiempo de comunicaciones necesario para las transferencias de datos hacia y desde el proceso *Master*.

Este algoritmo fue uno de los primeros incluidos en las distribuciones más conocidas de Linux, aunque algunas de ellas ya no lo incluyen más. Siempre es posible (y sencillo, de hecho) instalarlo en Linux, donde básicamente se pone en marcha el proceso timed a nivel de sistema operativo. En todos los casos la mejor resolución obtenida es de 1 milisegundo, a la que se llega bastante más rápidamente que con NTP. En principio, no cumple con uno de los objetivos originales de este trabajo que es llegar al orden de los microsegundos.

2.3 Algoritmo de Lamport

El algoritmo de Lamport es uno de los primeros propuestos para la sincronización de sistemas distribuidos [2] y se basa en la relación "sucede antes" más la utilización de los mensajes entre las computadoras como indicadores precisos de esta relación. Más específicamente, un mensaje no puede ser recibido antes de ser enviado y, por lo tanto, si se tienen marcas de tiempo de los envíos de los mensajes se puede verificar si el tiempo actual es coherente con la definición de la relación "antes de". Las tareas a llevar a cabo en cada computadora son relativamente sencillas, aunque afectan, en cierta manera, la forma en que se procesan los mensajes:

1. Se tiene un reloj local.
2. Cada vez que se envía un mensaje, se le *agrega* al mismo una marca de tiempo (*timestamp*) con el tiempo local del que envía.
3. Cada vez que llega un mensaje, se analiza la marca de tiempo del que envió, y
    a. Si la marca de tiempo es menor que el tiempo local, se asume que las computadoras están sincronizadas
    b. Si la marca de tiempo es mayor que el tiempo local, se cambia el tiempo local con la marca de tiempo del mensaje que se recibió más 1 (asumiendo, por ejemplo, que la transmisión necesita 1 unidad de tiempo)

La Fig. 1 muestra el avance del tiempo en dos computadoras, C1 y C2, y la forma en que la computadora C2 cambia su tiempo local a partir de la llegada de un mensaje con una marca de tiempo mayor que su tiempo local.

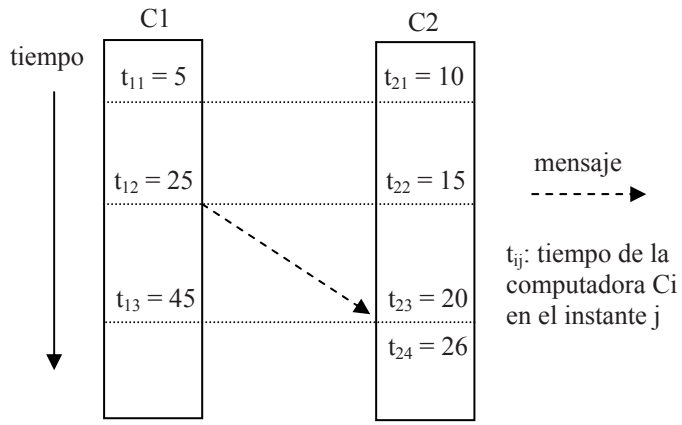


Figura 1: Ejemplo de Funcionamiento del Algoritmo de Lamport.

Este algoritmo soluciona el problema de la escalabilidad, pero no tiene un orden total de los eventos que suceden entre diferentes computadoras excepto los relacionados directamente con envío y recepción de mensajes. En el caso de la Fig. .1, no es posible determinar la relación de tiempos entre los eventos que suceden en la computadora C1 entre los instantes de tiempo t_{12} y t_{13} con los que suceden en la computadora C2 entre los instantes de tiempo t_{22} y t_{23}.

3 ANALISIS DE DIFERENCIAS ENTRE RELOJES

Los valores de hora en dos computadoras pueden diferir a partir de un instante de tiempo por un cierto error inicial (*offset*), e irá creciendo en el tiempo dado por las diferencias en sus frecuencias (primera derivada) más un error aleatorio dado por la deriva en el valor de frecuencia de cada una de las frecuencias, producto de cambios ambientales [15] [14]. Al menos la diferencia de dos relojes en un instante de tiempo puede ser estimado en una computadora a partir del tiempo de transmisión entre las dos máquinas. Si dicho tiempo fuese constante, se podria medir y eliminar este error utilizando la constante necesaria. Como ambos relojes no están sincronizados, el timepo debe ser medido por un solo reloj, por lo tanto se mide el tiempo de ida y vuelta de un paquete en la red de comunicaciones (*round trip time*). Se estima el tiempo de transmisión como la mitad del tiempo de ida y vuelta, lo cual no necesariamente es correcto. Una descomposición de este tiempo sería el de la Fig. 2, donde

- **Tiempo de envío**: Es el gastado en la construcción del paquete en las distintas capas hasta alcanzar la capa de acceso al medio (*MAC*). Es variable en función del estado del SO (cambio de contexto, planificación, etc.).
- **Tiempo de acceso al medio**: Depende del protocolo usado. En el protocolo Ethernet dependerá del tráfico en la red. Es aleatorio.
- **Tiempo de transmisión**: Depende de la velocidad de la placa de red y del largo del mensaje. Es determinístico.
- **Tiempo de propagación**: en una red local será despreciable ya que es lo que tarda en recorrer el cable entre placas de red - *switch* - placa de red más el retardo que introduce el *switch*.
- **Tiempo de recepción**: Es el tiempo que tarda en atravesar hasta la *MAC layer*. Es determinístico.
- **Tiempo de recibo**: Es el gastado en desarmar el paquete en las distintas capas hasta ser entregado a la aplicación. Depende del SO como el de envío.

Figura 2: Descomposición del Tiempo de un Mensaje entre Dos Computadoras.

Estos tiempos son difíciles de medir sin contar con equipamiento especializado de mediciones electrónicas. El principal problema que introducen referido a la sincronización de relojes es debido a que son variables, con lo que sólo se los puede tratar estadísticamente o directamente acotar. Otro enfoque es trabajar sobre los elementos de *hardware* y *software* (principalmente sistema operativo) que causan que sean variables. Sería deseable que aunque no se pueda eliminar el error, se pueda minimizar y además tener una estimación del mismo. Para ello se realizaron experimentos tendientes a medir la moda, valor mínimo y máximo en un ambiente como el que se especifica en los requerimientos. Para acotar aun más la parte variable, se midió el mínimo tiempo que tarda el sistema en acceder a los *buffers* de comunicación. A partir de estos experimentos, se determina que todas las trasmisiones se tomen como válidas solamente si fueron realizadas en un tiempo rtt

(*round-trip time*, tiempo de ida y vuelta de un paquete) igual a la moda. Con ello se tiende a acotar los errores debido a las comunicaciones tanto en la determinación del *offset* inicial entre relojes como de la diferencia de frecuencia. Un problema agregado es el de la asimetría entre los tiempos de ida y vuelta. Si se toman como válidas solo las referencias de tiempo trasmitidas en tiempo rtt igual al valor de la moda, es muy probable que dicha asimetría puede estar dada por la diferencia entre la moda y el mínimo, actualmente se está estudiando si es posible evitar el error por asumir que los tiempos de ida y vuelta de un mensaje son iguales.

Una vez resuelto el problema del *offset inicial*, se deben tener en cuenta las diferencias de frecuencias con las cuales se actualiza la hora en cada computadora. Los sistemas operativos como Linux usualmente tienen calculado un valor de frecuencia del oscilador a partir del cual actualiza la hora del sistema. Este valor, generalmente dado en MHz (millones de ciclos por segundo), no es necesariamente correcto para el registro de la hora con precisión de microsegundos. Por esta razón se considera adecuado hacer el cálculo de MHz del oscilador de cada computadora de manera tal que el cálculo de la hora posterior utilizando este valor tenga precisión del orden de microsegundos o mejor. Como se aclaró en el caso de NTP, tener un reloj local (o los relojes locales de todas las computadoras) con precisión del orden de microsegundos no necesariamente implica que el error de sincronización sea de esta magnitud.

El cálculo de MHz reales de una computadora es relativamente sencillo a partir de RDTSC y el uso del reloj de *hardware* del sistema. Aunque la precisión del reloj del sistema no sea muy apropiada para el registro de tiempo del orden de microsegundos, la frecuencia del mismo es suficientemente estable como para ser considerada constante en un intervalo de tiempo suficiente como para el cómputo inicial de MHz. Asumiendo que el oscilador con al que se accede con RDTSC es de frecuencia constante, el cálculo de MHz puede ser realizado en un único punto en el tiempo, con lo cual este tiempo inicial puede ser relativamente grande (segundos o quizás minutos). Sin embargo, el cálculo no necesariamente es tan sencillo en diferentes computadoras que luego se deben sincronizar. La razón fundamental está relacionada con el reloj de hardware de referencia con el cual se realiza el cálculo de MHz de cada computadora.

La alternativa es, justamente, que una sola de las computadoras, usualmente la que funciona como servidor de hora, proporcione las referencias de hora con la cual se hace el cálculo de MHz en cada una de las demás computadoras. Este esquema es sencillo y también se pueden evitar errores haciendo mayor el tiempo durante el cual se contabilizan las oscilaciones en base a las cuales se lleva a cabo el cálculo de MHz. En cierta forma, tanto las referencias de tiempo del reloj de hardware de una PC como las referencias de tiempo que se reciben desde otra computadora tienen un margen de error, que se puede disminuir proporcionalmente con el propio tiempo que se contabiliza. En este caso también se utiliza la red de interconexión y se toman los tiempos de mensajes como significativos para el control del error en el cálculo de MHz. Más específicamente:

- En las computadoras utilizadas de 2.4 GHz se tienen aproximadamente 2.4×10^9 oscilaciones por segundo.
- Un paquete TCP utiliza aproximadamente 120 μs como tiempo de ida y vuelta, con lo cual se puede asumir que se necesitan aproximadamente 60 μs para un envío-recepción de un paquete TCP individual (o *one-way transmission time*).
- Durante un tiempo de 10 segundos (10^7 μs) en el servidor, el tiempo que se contabiliza en los clientes a partir de los mensajes recibidos podría tener el error de tiempo de transmisión dos mensajes, es decir 120 μs, con lo cual el posible error está acotado por $120/10^7$, es decir que el error en la contabilización de oscilaciones es de $120/10^7$, o de otra manera, si k es la cantidad de oscilaciones contabilizadas, el valor real puede variar $\pm120/10^7\times k$.

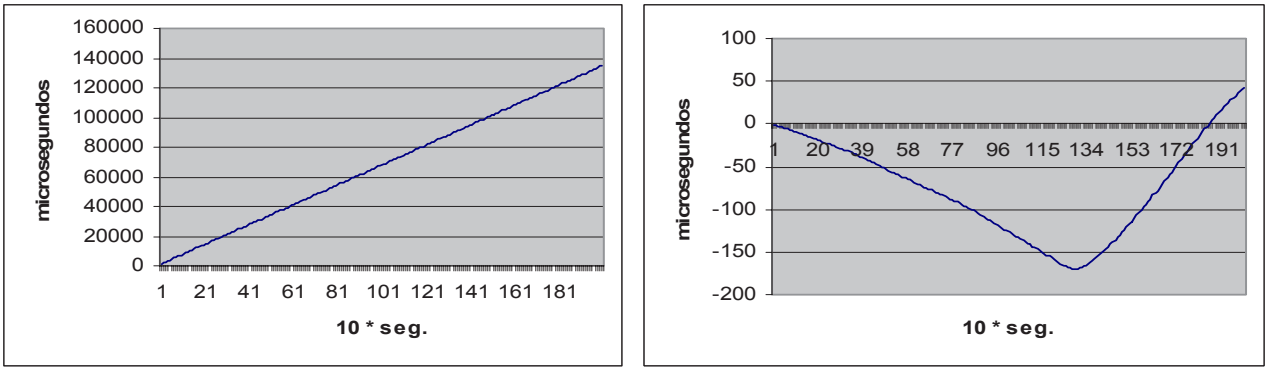
La Tabla 2 muestra los valores que se obtienen al sincronizar relojes con cálculo de MHz totalmente independientes (con los relojes locales) y con cálculo de MHz con referencia única, es decir donde una sola computadora proporciona las referencias de tiempo para el cálculo de MHz. El experimento se llevó 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. La Tabla 2 muestra los errores mínimo y máximo de sincronización y debe recordarse que el error de sincronización se define como la diferencia en los tiempos locales de las computadoras y estos tiempos, a su vez, tienen una resolución de aproximadamente 1.18 μs [20].

Tabla 2: Errores de Sincronización por Cálculo de MHz.

Cant. Máq.	**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

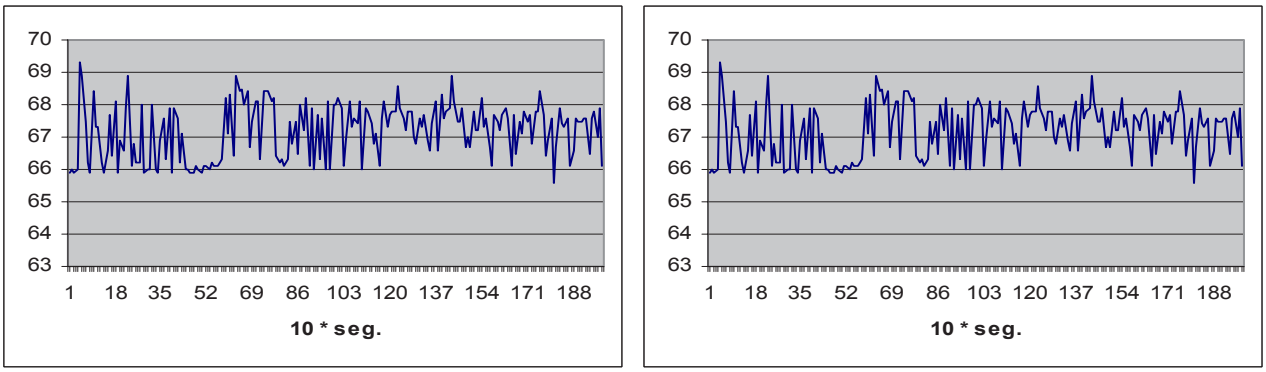
En la Fig. 3 se puede ver la diferencia de hora a partir de un ajuste inicial entre dos máquinas cuya frecuencia de reloj ha sido corregida tomando referencia local y referencia única. Se debe notar la diferencia en cuanto a los valores de las diferencias: para la Fig. 2a) varía entre 0 y poco menos de 150000 μs, y para la Fig. 2b) varía entre aproximadamente -150 y 50 μs. Obsérvese que con referencia local, Fig. 2a), la diferencia aumenta proporcionalmente a medida que transcurre el tiempo y en aproximadamente media hora sin resincronizar, se ha apartado del valor de la hora de la máquina de referencia alrededor de 0,14 segundos. En el caso de referencia única, Fig. 2b) al transcurrir el mismo tiempo la diferencia máxima fue 0,000150 segundos (150 μs.). Es de hacer notar que en vez de aumentar constantemente, vuelve a converger al valor de la hora de referencia. Estas variaciones corresponden a variaciones de tipo ambiental, como podrá verificarse en la primera y segunda derivada. La Fig. 4 muestra los valores relacionados con la primera derivada de los valores de hora. En el caso de referencia local, Fig. 4a), este valor es relativamente grande y siempre positivo, lo cual implica el constante aumento de las diferencias de tiempo. En el caso de referencia única, el cambio en el signo hace converger las diferencias de hora hacia valores menores. Como dato adicional, se grafican en la Fig. 5 los valores de la segunda derivada, donde se observa que tanto para referencia local como única son similares (0 a 0,3). Esta derivada es la que

muestra los cambios debidos a factores ambientales, que normalmente son difíciles de controlar más alla de recomendaciones tales como tener las máquinas en el mismo cuarto o en condiciones climáticas similares, con las medidas de refrigeración acordes al hardware, etc.



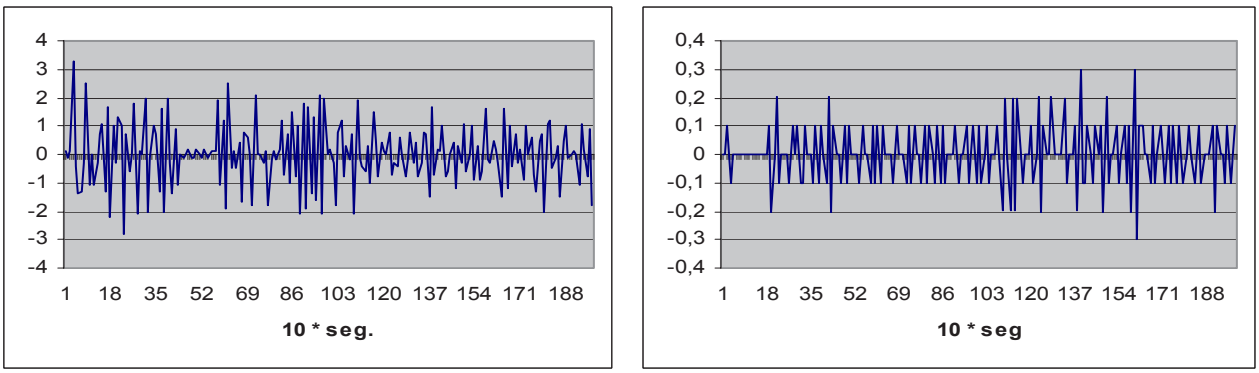
a) Referencia Local b) Referencia Unica.

Figura 3: Diferencias de Tiempo Dependiendo del Cálculo de MHz.



a) Referencia Local b) Referencia Unica.

Figura 4: Primera Derivada de las Diferencias de Tiempo Dependiendo del Cálculo de MHz.



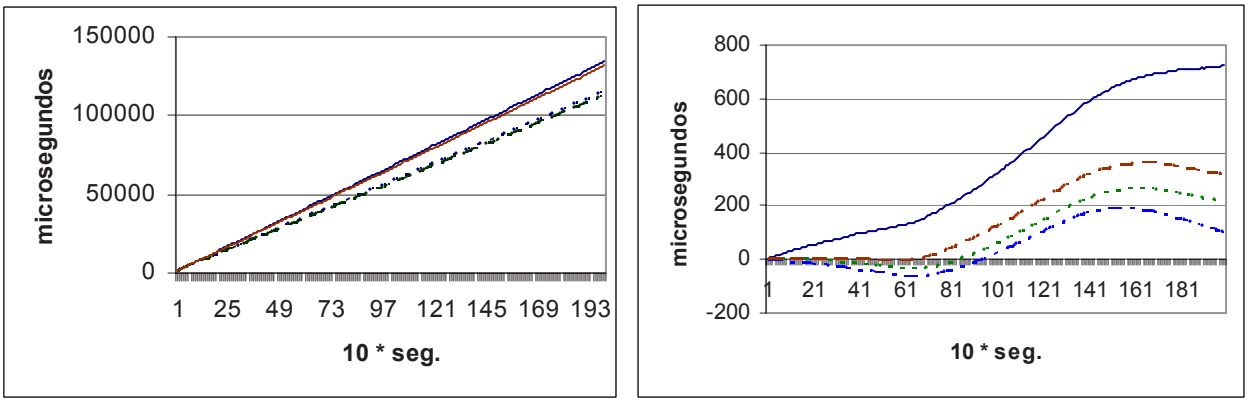
a) Referencia Local b) Referencia Unica.

Figura 5: Segunda Derivada de las Diferencias de Tiempo Dependiendo del Cálculo de MHz.

En todos los experimentos se tienen en cuenta las diferencias entre relojes, ya que para ver solamente las diferencias del cliente, el servidor debería proporcionar un reloj perfecto. Una posible mejora sería disponer de una máquina con sistema operativo de tiempo real para suministrar la

referencia de tiempo. Otra forma adicional de mejorar los valores a medir sería tener un reloj de referencia mejor. Como la idea es no recurrir a ningun tipo de hardware adicional, una de las posibilidades es armar un ensamble de relojes. Se demuestra que un arreglo de relojes puede dar una hora más estable [21][1] y también puede tomar la idea del algoritmo de Berkeley [2].

En la Fig. 6 se pueden ver las diferencias de hora de 4 clientes con respecto al servidor. En el caso de referencia local, Fig. 6a), luego de 2000 segundos sin resincronizar el desfasaje máximo supera los 30000 microsegundos. Con referencia única, Fig. 6b) dicho valor es de 750 microsegundos, o sea unas 40 veces más pequeño. Esto demuestra que, más allá del error inicial que puede ser aproximadamente igual en ambos casos, se obtiene un error en frecuencia sensiblemente menor, lo que llevará a una menor pérdida de sincronización a medida que transcurre el tiempo, evitando una resincronización de relojes para mediciones de tiempos relativamente largos. El hecho que un grafico presenta rectas y el otro curvas está relacionado con la escala del eje y, en realidad la recta no es tal, sino su derivada segunda sería 0.



a) Referencia Local b) Referencia Unica.

Figura 6:Diferencias de Tiempo de 4 Clientes con Respecto al Servidor de Hora.

4 CONCLUSIONES

La utilización de una única referencia de tiempo para la corrección del error de frecuencia en las diferentes máquinas en lugar del propio reloj local presenta una reducción importante de los errores remanentes. El algoritmo planteado, una modificación al algoritmo de Cristian [3] [5], ha demostrado su eficacia en las pruebas en un cluster. En un ambiente fuera de red local los errores pueden ser demasiado grandes respecto de las mediciones que se deban realizar debido a la variabilidad de los tiempos de comunicación.

Con respecto a la escalabilidad, el modelo cliente servidor utilizado puede presentar problemas, una posible solución sería probar con comunicaciones *broadcast* para la sincronización. De hecho, se deberían estimar/evaluar los problemas de escalabilidad y en función de los valores observados proponer una posible solución. También sería necesario evaluar la sobrecarga que se impone en el servidor específicamente con respecto a la cantidad de clientes, algo que también está relacionado con la escalabilidad del esquema de sincronización planteado en este trabajo.

Como extensiones futuras, siempre es deseable la sincronización externa de los relojes [5]. 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

REFERENCIAS

[1] Agilent Application Note AN 1289, "The Science of Timekeeping"

[2] G. Coulouris, J. Dollimore, T. Kindberg, “Sistemas Distribuidos. Conceptos y Diseño”, 3ª edición. Pearson Educación, 2001. ISBN: 8478290494.

[3] F. Cristian. "Probabilistic Clock Synchronization". Distributed Computing, 3: 146–158, 1989.

[4] K. J. Elson, L. Girod, D. Estrin, "FineGrained Network Time Synchronization using Reference Broadcasts", Proceedings of fifth symposium on Operating System Design and Implementation. December 2002.

[5] C. Fetzer, F. Christian, "Integrating External and Internal Clock Synchronization", June 1996.

[6] D. A. Grove. "Performance Modelling of messagepassing parallel programs", May 2003.

[7] R. Gusella, S. Zatti "An Election Algorithm for a Distributed Clock Synchronization Program" EECS Department University of California, Berkeley Technical Report No. UCB/CSD-86-275 1986

[8] K. H. Kim, C. Im, P. Athreya, "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.

[9] D. L. Mills, "A Brief History of NTP Time: Confessions of an Internet Timekeeper". ACM Computer Communications Review 33, 2 (April 2003), pp 922.

[10] D. L. Mills, "Improved algorithms for synchronizing computer network clocks", IEEE/ACM Transactions on Networks June 1995.

[11] D. L. Mills, "Measured performance of the Network Time Protocol in the Internet System". ACM Computer Communication Review 20, Jan. 1990. pp. 6575.

[12] D.L. Mills, “Network Time Protocol (Version 3) specification, implementation and analysis”. Network Working Group Report RFC-1305, University of Delaware, March 1992, 113 pp.

[13] D.L. Mills, "Internet time Synchronization: the Network Time Protocol", IEEE trans. Communications COM39, October 1991, pp. 14821493.

[14] D. L. Mills, "Modelling and analysis of computer network clocks", Electrical Engineering Department Report 9252, University of Delaware, May 1992.

[15] D. L. Mills, “Precision synchronization of computer network clocks”, ACM Computer Communication Review 24, 2 (April 1994). 28-43.

[16] D. L. Mills, P. H. Kamp, "The Nanokernel", Proc. Precision Time and Time Interval (PTTI) Applications and Planning Meeting (Reston VA, November 2000).

[17] D. L. Mills, "Unix kernel modifications for precision time synchronization". Electrical Engineering Department Report 94101, University of Delaware, October 1994.

[18] S. Mishra, C. Fetzer, F. Cristian, “The Timewheel Asynchronous Atomic Broadcast Protocol”, PDPTA 1997: 1239-1258

[19] 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

[20] F. L. Romero, F. G. Tinetti, "Sincronización de Relojes en Ambientes Distribuidos" IX Workshop de Investigadores en Ciencias de la Computación,Trelew, Mayo 2007.

[21] D.B. Sullivan, D.W. Allan, D.A. Howe, and F.L. Walls, "Characterization of Clocks and Oscillators", NIST Tech Note 1337, 1990. (BIN: 868)

[22] P. Work, K. Nguyen, "Measure Code Sections Using The Enhanced Timer", http://www.intel.com/cd/ids/developer/asmo-na/eng/209859.htm, October 2005.