Trabajo Preliminar para la Obtención de Tiempos Sincronizados en Clusters con Nodos de Múltiples Núcleos

Fernando Romero, Horacio Villagarcía[1], Fernando G. Tinetti[2]
Instituto de Investigación en Informática III-LIDI
Facultad de Informática, UNLP
{fromero, hvw, fernando}@lidi.info.unlp.edu.ar

Resumen. En este artículo se presenta una extensión de la metodología y herramienta para instrumentación de programas paralelos en plataformas de cómputo distribuidas que se venía elaborando en trabajos anteriores. Específicamente, la extensión contempla el hardware de múltipes núcleos que actualmente se utiliza en los nodos individuales de los clusters. En el contexto de instrumentación de tiempo en ambientes distribuidos es fundamental la sincronización de los relojes que intervienen. Se mantiene el algoritmo básico de sincronización, usando las estrategias clásicas que se utilizan en ambientes distribuidos para entornos de cluster, con 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.

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

1 Introducción

En este trabajo se continúa la línea de investigación sobre sincronización de relojes de computadoras, con fines de realizar instrumentación [5] [6] [7] [8] [9] [10] [11] [12]. Se trata de extender la herramienta presentada oportunamente a fin de incluir los sistemas con múltiples núcleos. En estos sistemas, las estrategias utilizadas en los sistemas monoprocesador presentan inconvenientes. La referencia de tiempo utilizada, proporcionada originalmente por el dispositivo de hardware "TimeStamp Counter" (TSC), tiene dos limitaciones:

1) Cada núcleo posee su propio TSC, independiente y a priori sin sincronización respecto a los demás.
2) Es posible que los sistemas de ahorro de energía alteren la frecuencia con la que se actualiza la referencia en el TSC. Es decir que, a priori, no se puede asumir que la frecuencia para la actualización del TSC sea constante (más allá de las características físicas del propio TSC y del ambiente).

1 Profesional Principal CICPBA
2 Investigador CICPBA

Estas limitaciones han llevado a buscar una alternativa de referencia de tiempo que, sin dejar de cumplir con los requerimientos en los que se basaba el diseño de la herramienta de instrumentación [8], no presente estos inconvenientes. Los requerimientos originales son:

- Una herramienta que pueda ser usada inicialmente en un cluster de PCs, 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 para realizar los análisis de rendimiento que sean necesarios.
- 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.

Analizando los últimos cambios en los dispositivos periféricos, se contempla la utilización de los registros temporizadores "*High Precision Event Timers*" (HPET) [4] como referencia de tiempo. En primera instancia se cumple con los requerimientos anteriores y provee una referencia única para todos los núcleos de un mismo nodo de cada cluster. Evidentemente, nodos diferentes son independientes y deben ser sincronizados, para lo cual en principio se seguirá con la metodología/algoritmo estándar.

2. Hardware y HPET

Las especificaciones industriales en los manuales dedicados a los usuarios OEM y proveedores de BIOS que usen series de chipset Intel para basar sus productos [3] describen la "Platform Controller Hub" (PCH) como término genérico que engloba las funciones y capacidades para soporte de E/S. En esos manuales se detallan las características de las señales (temporales y eléctricas), empaquetado, mapeo de memoria y registros así como las distintas interfaces y sus registros de configuración. Entre las funciones y capacidades tradicionalmente utilizadas encontramos las especificaciones para PCI, USB, SATA, PCI Express y LAN. Se agregan además nuevas capacidades entre las que se encuentra hardware adicional de temporizadores para complementar y eventualmente reemplazar las funciones de generación de intervalos de tiempo y de interrupciones periódicas provistas por el 8254 Programmable Interval Timer y el Real Time Clock en las computadoras personales basadas en arquitecturas Intel (IA-PC). Los temporizadores de eventos de alta precisión (en *inglés High Performance Event Timers* – HPET) son definidos [4] como un conjunto de registros de temporización separados en bloques que el sistema operativo puede utilizar e inclusive en el futuro podrá asignar específicamente para ser utilizados en forma directa por una aplicación. Se propone su utilización en

sincronización de audio y video, en programación (*scheduling*) de tareas, hilos o procesos por generación de interrupciones y operaciones de marcado de tiempo en plataformas multiprocesador.

2.1 Registros Temporizadores

Cada temporizador o *timer* puede ser configurado para generar una interrupción separada. La arquitectura permite bloques de 32 timers, pudiendo soportarse hasta 8 bloques, dando un total de 256 timers. Se establece el soporte de 3 timers como mínimo. Los timers son implementados con un único contador ascendente (*main counter*) y un conjunto de comparadores y registros de comparación (*match register*). El contador ascendente se incrementa de forma monotónica, es decir cuando se realizan dos lecturas consecutivas del contador, la segunda lectura siempre tiene valores mayores que la anterior salvo el caso límite de desborde (*roll over*). Cada timer incluye un comparador y un registro de comparación. Cada timer puede generar una interrupción cuando el valor en su registro de comparación es igual al valor del contador ascendente. La interrupción generada por cada timer poseerá características definidas en un registro asociado para configuración y ruteo en el que se puede por ejemplo, habilitar la generación de esa interrupción en forma periódica. Los registros asociados con los timers se asignan al espacio de memoria de modo similar que el controlador avanzado de interrupciones de entrada/salida (I/O APIC, *Advanced Programmable Interrupt Controller*). Sin embargo, no se implementan como una función estándar PCI. El BIOS reporta al sistema operativo la ubicación asignada en memoria para el bloque de los timers. No se espera que el sistema operativo cambie la ubicación de estos timers una vez que se establece por/en el BIOS.

2.2 Mapeo en Memoria

La CPU puede acceder directamente a cada registro timer porque éstos están mapeados en memoria. El espacio de registros timer es de 1024 bytes. Son alineados en límites de 64 bits para simplificar su implementación con procesadores de 64 bits. El modelo de registros permite que cada bloque timer contenga hasta 32 timers, donde cada timer consiste de un comparador más un registro de comparación. La Figura 1 muestra una visión genérica de los registros timer y su mapeo en memoria. En ella debe observarse que el funcionamiento del bloque de HPET, se define en tres registros generales (de capacidades e identificación, de configuración y de generación de interrupciones) y un único contador ascendente. Estos están ubicados a partir de la dirección base del bloque, en los desplazamientos 0H, 10H, 20H y F0H respectivamente. En el primero de los mencionados registros (registro de capacidades generales e identificación) se encuentran definidos el período con el que trabaja el contador ascendente y la cantidad de timers implementados en el hardware. En las posiciones de memoria a continuación de las anteriores se encuentran en oren sucesivo los registros asociados a cada timer comenzando con el timer 0. Por cada timer se tienen 3 registros de 64 bits (de configuración, de comparación y de ruteo de interrupciones) que se ubican cada 32 bytes. La ubicación relativa de los registros de

cada timer puede determinarse siguiendo la fórmula (20*n+100H) donde n es el número de timer.

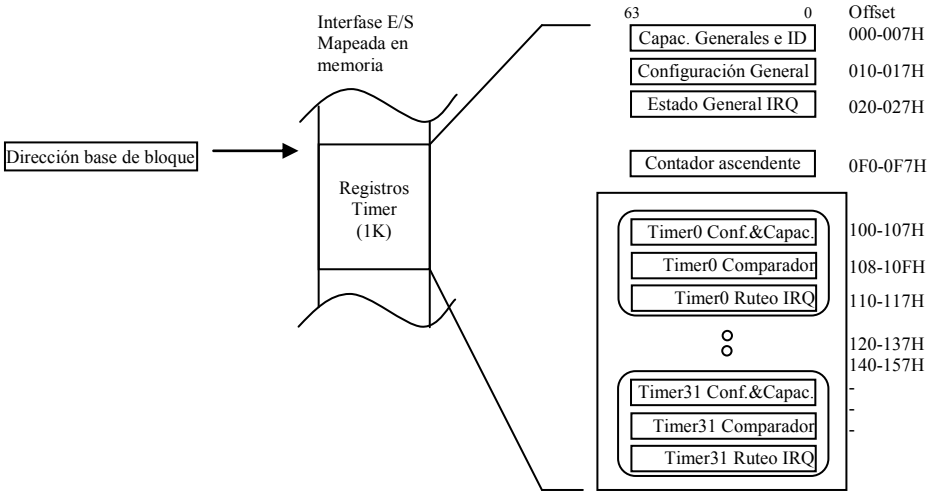

Figura 1. Modelo de Registros y su Ubicación Relativa en Memoria

3 El Problema de la Sincronización de Relojes

Como en la gran mayoría de los sistemas distribuidos, en un cluster no existe la posibilidad de acceso de todos los nodos a un único reloj. Por ello, se debe implementar la posibilidad de acceder al propio reloj local de cada nodo, y mantener los relojes locales sincronizados con respecto al reloj de un servidor (sincronización interna) [2]. La sincronización de relojes de computadoras en un ambiente distribuido se basa en acotar las diferencias de tiempo en un instante dado. Para ello se debe comunicar una referencia inicial a todas las máquinas. A partir de dicha referencia, se debe estimar la deriva de los relojes individuales, a fin de corregirla. Para ello se hace necesario el intercambio de más referencias de tiempo entre servidor y cliente. La comunicación de estas referencias, a través de la red de comunicaciones subyacente, encuentra un problema en la variabilidad de los tiempos de comunicaciones. Esto se acota utilizando un algoritmo basado en los métodos estadísticos de los tiempos de comunicaciones [1] [2]. En este algoritmo se realiza una estadística de los tiempos de comunicaciones de mensajes entre los diferentes nodos. Establecido el tiempo mas frecuente (tiempo moda), sólo se consideran válidos los mensajes que lleguen en este tiempo. La validación del tiempo de referencia se realiza en el nodo que envía el mensaje al arribar el aviso de reconocimiento y calcular el tiempo RTT (*Round Trip Time*). En la Figura 2 vemos que la referencia llegará con una demora D = RTT/2, si estimamos que los tiempos de ida y de vuelta son iguales (ó simétricos), lo cual es altamente probable en los ambientes de cluster y redes locales. La variabilidad en la simetría de los tiempos de comunicaciones es parte de los errores de sincronización que efectivamente se tendrán en la implementación final.

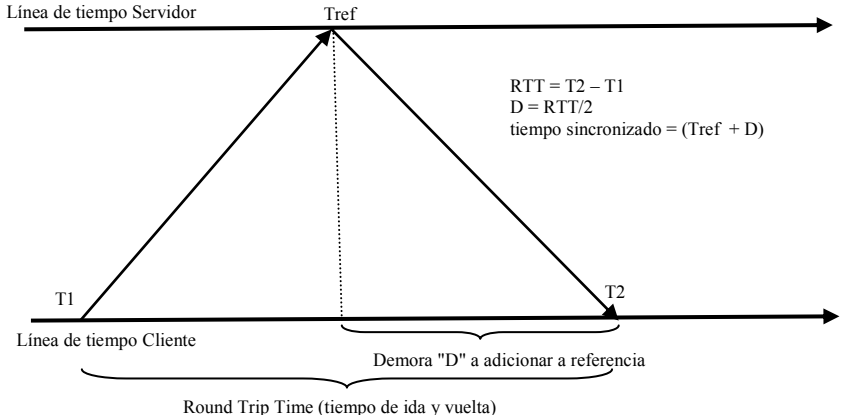

Figura 2: Tiempos de comunicación de una referencia entre nodos

4 Experimentos para Determinar Sobrecarga y Resolución

En la instrumentación de aplicaciones distribuidas se pretende mejorar su rendimiento a partir del análisis de la sucesión de eventos en tiempo de ejecución. Para el análisis correcto de la sucesión de eventos de una aplicación paralela es necesario contar con una referencia horaria única, o al menos con diferencias acotadas, en los nodos del cluster. Se debería tener en cuenta que la utilización de dicha referencia no cause diferencias de rendimiento (o funcionamiento, en general) significativas de la que tendría la aplicación sin instrumentar (intrusión). Se realizaron experimentos a fin de determinar sobrecarga y resolución de las diferentes fuentes de hora, a fin de comprobar que se cumplan los requerimientos al respecto. Todos los experimentos se llevaron a cabo en un cluster compuesto por nodos homogéneos, formados por procesadores Intel XEON E5405 2GHz con 2GB RAM y Sistema Operativo Linux 2.5.32-5.

En la Tabla 1 se muestran los resultados estadísticos de los tiempos de sobrecarga de diferentes fuentes de suministro de hora local, en microsegundos.

Tabla 1. Sobrecarga de Diferentes Fuentes de Tiempo.

Fuente	Máx.	Mín.	Prom.
gettimeofday	0,980918	0,674944	0,79087915
HPET	0,989917	0,506958	0,5185626
TSC	0,614949	0,059995	0,07464648

La Figura 3 muestra los histogramas de frecuencia de sobrecarga de cada uno de los métodos que pueden proporcionar una referencia de hora local. Se puede apreciar que tanto en los valores de tiempos de lectura del HPET como en los del TSC los

valores se concentran cerca del mínimo, mostrándose los de gettimeofday más dispersos, probablemente debido a que se trata de una llamada al sistema, con lo que intervienen varios factores que determinan su tiempo de ejecución. Si bien el menor tiempo de sobrecarga se obtiene con TSC, se recuerda que se utiliza HPET para disponer de una referencia única para todos los núcleos de todos los procesadores de cada nodo.

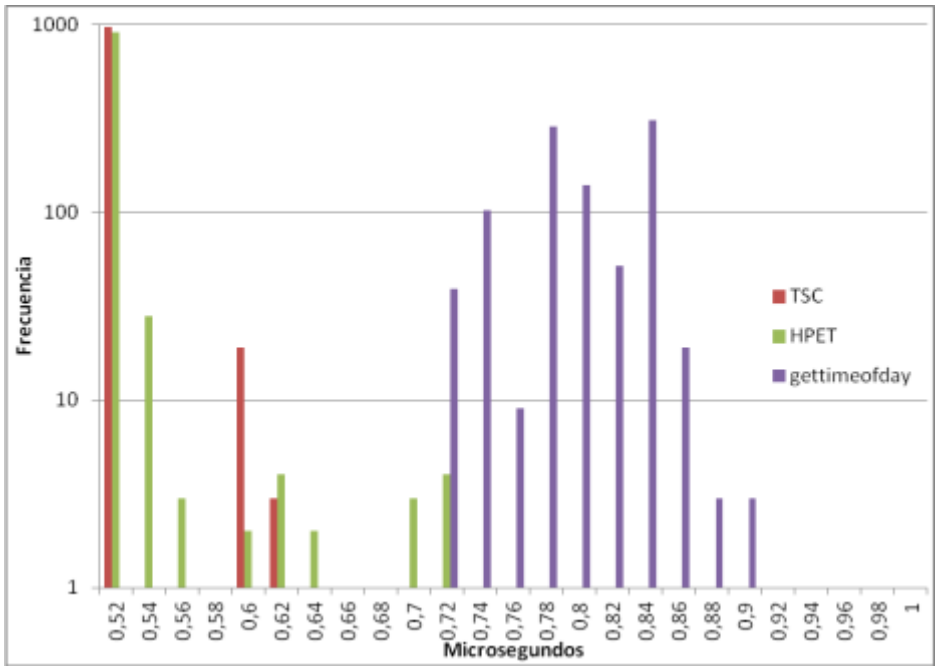

Figura 3. Distribución de Tiempos de Sobrecarga de gettimeofday, HPET y TSC.

En cuanto a la resolución, en la mayoría de los sistemas el contador del HPET se actualiza a una frecuencia del orden de los 14Mhz, pero en el caso de hacer mediciones sucesivas a máxima velocidad, se obtiene un tiempo entre mediciones de 0,55 microsegundos, lo que pudiera limitar a este valor la resolución. Aún en este valor, la resolución cumple con el requerimiento de ser menor o igual a 1 microsegundo.

5 Conclusiones y Trabajos Futuros

Los sistemas multiprocesadores continúan evolucionando, con lo que se deben seguir investigando formas de disponer de una referencia de tiempo en las máquinas de este tipo, que cumpla con los requerimientos enunciados anteriormente. En este trabajo se ha demostrado la posibilidad de utilización del HPET y de acuerdo con los resultados, la sobrecarga se puede considerar dentro de los límites de lo aceptable. Todo indica que teniendo una referencia de tiempo única para todos los núcleos de los posibles

múltiples procesadores de un nodo se podrían sincronizar todos los nodos de un cluster. Como es usual, el trabajo inmediato para continuar con esta línea de investigación será la implementación de la sincronización en todo un cluster completo. Esta implementación deberá contemplar los límites (errores) de la sincronización.

Una vez implementada la sincronización, se podrá efectivizar su uso en la evaluación de rendimiento de aplicaciones paralelas en clusters. En todos los casos se deberá tener en cuenta la red de interconexión que no solamente se utilizará para sincronización sino también para la propia aplicación paralela. Como hasta ahora, se mantendrá un uso controlado y conocido a priori de la red de interconexión. Siempre que la red mejore en rendimiento se podría mejorar también en términos de reducción de las cotas de error de sincronización.

Bibliografía

1. Christian F. "Probabilistic Clock Synchronization", Distributed Computing, 3: 146–158, 1989.
2. Fetzer C., Christian F., "Integrating External and Internal Clock Synchronization", Journal of Real-Time Systems, Vol. 12, Issue 2 pp. 123-171 (1997)
3. Intel Corporation, Intel® C600 Series Chipset Data Sheet, Document Number: 326514-001, March 2012.
4. Intel Corporation, IA-PC HPET Specification, Revision 1.0a (2004)
5. F. L. Romero, A. E. De Giusti, F. 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, Univ. Nac. de Chilecito, Chilecito, Argentina, Octubre 2008. ISBN 978-987-24611-0-2.
6. F. G. Tinetti, F. L. Romero, A. E. De Giusti "Clock Synchronization in Clusters for Performance Evaluation: Numeric/Scientific Computing". Proc. 2009 World Congress on Computer Science and Information Engineering, IEEE Computer Society, March 2009, Los Angeles, USA, ISBN 978-0-7695-3507-4/08.
7. F. Romero, "Sincronización de Relojes en Ambientes Distribuidos", Tesis de Maestría en Redes de Datos, Fac. de Informática, UNLP, Mayo de 2009.
8. F. G. Tinetti, F. L. Romero, A. E. De Giusti "Evaluación de la Sincronización Periódica de Relojes de Computadoras para Instrumentación". XV Congreso Argentino de Ciencias de la Computación (XV CACIC) Univ. Nac. de Jujuy, San Salvador de Jujuy, Argentina, octubre 2009. ISBN 978-897-24068-4-1. pp.301-310. (2009)
9. F. G. Tinetti, F. L. Romero, "Herramientas para Instrumentación de Programas Paralelos en Ambientes Distribuidos", XII Congreso Argentino de Ciencias de la Computación, Univ. Nac. de San Luis, Potrero de los Funes, San Luis, Argentina, Octubre 2006. ISBN 950-609-050-5. pp.1414-1423
10. Fernando L. Romero, Fernando G. Tinetti. "Sincronización de Relojes en Ambientes Distribuidos" IX Workshop de Investigadores en Ciencias de la Computación. WICC 2007. Facultad de Ingeniería – UNPSJB - Trelew – Chubut. 3 y 4 de Mayo de 2007. ISBN 978-950-763-075-0. pp. 638-642. Congreso Nacional. Referato Nacional.
11. Work P., Nguyen K., "Measure Code Sections Using The Enhanced Timer", http://www.intel.com, October 2005.
12. 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.