Análisis del impacto del uso de TLBs en SoCs desde la perspectiva de la problemática de los *TLB shootdowns*

Pablo. Suárez Reyero[1]

[1] *Universidad Complutense de Madrid, Arquitectura de Sistemas Integrados.*

Fecha: Marzo del 2023

Abstract— En este informe de 3 páginas queda recogido el análisis sobre la influencia de una TLB implementada en una arquitectura determinada. Cabe destacar que este análisis se ha realizado desde el punto de vista de la problemática de los *TLB SHOOTDOWNS* . Aunque se han recurrido ha otros *papers* con el fin de recabar más información sobre el impacto de las TLBs en los sistemas embebidos.

Keywords— TLB, TLB SHOOTDOWNS, RENDIMIENTO.

Para realizar este análisis me he basado en el siguiente *paper*: `https://ieeexplore.ieee.org/document/6113842`

I. Introducción

Los Búferes *Lookaside* de traducción, del inglés *(Translation Lookaside Buffers)*, i.e: TLB, se emplean habitualmente en diseños de procesadores actuales y tienen un impacto notable en el rendimiento global del sistema. En trabajos de investigaciones anteriores se ha estudiado exhaustivamente el diseño de este tipo de memorias, con el fin de reducir las tasas de fallo, así como los tiempos de acceso, especialmente en monoprocesadores [1]. Con el creciente predominio de los *chip-multiprocessors* (CMPs)[2], es necesario examinar el rendimiento de las TLBs en el contexto de las cargas de trabajo paralelas ([1]).

En este análisis se ha hecho énfasis en una de las problemáticas más comunes que surge como consecuencia del auge de los CMPs con una TLB por core, los *TLB shootdowns*. Los CMPs actuales se basan en el OS para aproximar el conjunto de TLBs que almacenan en caché un mapeo y sincronizar las TLBs mediante Interrupciones Inter-Procesador (IPIs) y controladores de software [2]. Aunque mi análisis está focalizado en la problemática comentada anteriormente, considero que es un análisis importante, ya que el uso de los CMPs se ha popularizado, y entender el impacto de esta problemática generada por las TLBs en este tipo de arquitecturas, es relevante dentro del mundo de los procesos, *threads*, etc.

Teniendo esto en cuenta, se han analizado las conclusiones y resultados expuestos en el *paper* [1]: *"DiDi: Mitigating The Performance Impact of TLB Shootdowns Using A Shared TLB Directory"*, caraterizando el impacto de los *TLB shootdowns* en el rendimiento y escalabilidad de los multiprocesadores.

Una cosa clara, que se verá más adelante, es como el coste y la frecuencia de los *TLB shootdowns* se ven incrementados cuando el número de procesadores aumenta.

II. Metodología

Antes de analizar los resultados expuestos en el *paper* mencionado, se introduce y explica muy brevemente la problemática en la siguiente sección. Una vez metidos de lleno en la cuestión, se analiza el impacto que tiene en el resto del sistema. A lo largo de las conclusiones, se adjuntan gráficas extraídas del articulo analizado que ilustran el impacto en diferentes arquitecturas (Intel y AMD).

III. TLB SHOOTDOWNS

Este concepto puede entenderse fácilmente con el siguiente ejemplo sacado de `https://stackoverflow.com/questions/3748384/what-is-tlb-shootdown`:

- Suponemos que hay una parte de memoria compartida por todos los procesadores del sistema.
- Uno de los procesadores restringe el acceso a una página de esa memoria compartida.
- Ahora, todos los procesadores tienen que vaciar sus TLBs, de modo que los que podían acceder a esa página ya no puedan hacerlo.

Las acciones de uno de los procesadores que causan que las TLBs se vacien en otros procesadores es lo que recibe el nombre de *TLB shootdown*. Esto tiene consecuencias en el

[1] Un sistema monoprocesador se define como un sistema informático que tiene una única unidad central de procesamiento

[2] Un chip multiprocesador (CMP) es una arquitectura en la que se integran varias unidades de procesamiento en un único circuito integrado o en varias matrices de un mismo paquete.

rendimiento del procesador, que se traducirá en una mayor latencia.

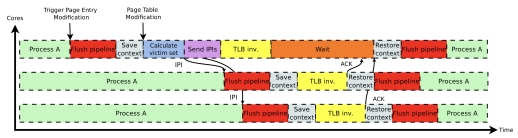

Fig. 1: TLB SHOOTDOWN

En la Figura 1 se muestra el *timeline* del proceso. Naturalmente, esto afecta directamente al SoC aumentando la latencia.

IV. Impacto de los TLB shootdowns

Para poder conseguir un *mapping* correcto entre direcciones virtuales y físicas, se requiere de un número de accesos a memoria proporcionales a la profundidad de la tabla de páginas (Intel y AMD utilizan tablas de páginas de 4 niveles). Este *mapeo* que se establece entre los dos tipos de direcciones, es critico en cuanto al rendimiento del procesador se refiere, ya que esto tiene lugar en cada operación de memoria. Por esta razón se utilizan las TLBs, para almacenar en una memoria *on-chip* la información sobre la traducción de direcciones, con el fin de no penalizar tanto a la η_{CMP}.

Debido a la falta de una "buena" estructura *hardware*, y a que las TLBs son solo estructuras de lectura, estas han de mantener una cierta coherencia a nivel de software (i.e: mantener una visión consistente de la memoria virtual, que todos vean lo mismo) esto lo hacen mediante los IPIs *Inter-Processor Interrupts* (IPIs) en el proceso explicado antes (i.e: los *TLB shootdowns*). Los *overheads* asociados al procesamiento de interrupciones hacen que los *TLB shootdowns* sean un cuello de botella en el rendimiento, impidiendo así la escalabilidad[3] de los multiprocesadores. El *paper* analizado se centra en la explicación de la solución a este problema: DiDi [4].

El coste de los procesos *TLB shootdowns* afectan a la carga de trabajo de los multiprocesadores e impiden el desarrollo de modelos de programación paralelos de alto nivel cada vez más presentes, aunque en estos modelos las modificaciones frecuentes de la tabla de páginas hacen que estos procesos sincronicen todas las TLBs del sistema, impactando negativamente al conjunto. Pero, si se reduce el *overhead* asociado a dichos procesos, se podría potenciar la escalabilidad de esos modelos de programación.

Teniendo en cuenta todo lo anterior, concluimos con que lo ideal es tener un soporte hardware para los *TLB shootdowns* con overheads pequeños.

En la Figura 2 se muestra la latencia de un solo *TLB shootdown* tanto para el iniciador [5] como para los esclavos

[3]La escalabilidad es la propiedad de un sistema de gestionar una cantidad creciente de trabajo añadiendo recursos al sistema.

[4]DiDi es el acrónimo de Dictionary Directory, que es el servicio semántico proporcionado por la TLB de segundo nivel de los cores de procesamiento

[5]Es el core que ejecuta una operación que modifica la tabla de páginas, esto hace que el SO bloquee la entrada en la tabla de páginas.

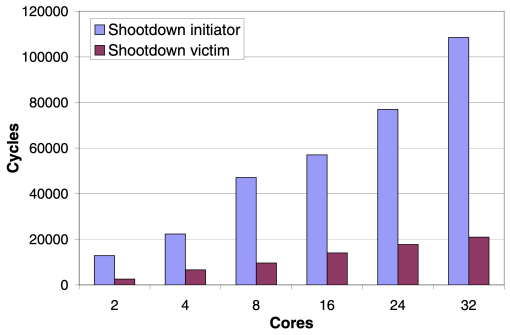

Fig. 2: Latencia por *shootdown*, Romanescu et al [3]

[6] en función del número de procesadores implicados en el *shootdown*. En lo que a este análisis respecta, he obviado los dos tipos de *shootdowns* mostrados en la figura, porque lo importante es el comportamiento lineal que sigue el número de ciclos según aumenta el $\#_{\text{cores}}$. Este aumento lineal de ciclos se traduce en un aumento lineal de la latencia. Cabe destacar que esta latencia no contempla los efectos secundarios de los *TLB shootdowns* (i.e: *overheads*), como la invalidación de la TLB, que resulta en un mayor número de ciclos. Según he ido analizando resultados y conclusiones, uno puede darse cuenta como, el impacto de las TLBs en μarquitecturas es notable, como la latencia del SoC y su área. Como consecuencia del aumento de la latencia, su consumo también lo hace, decrementando así la η del sistema. Como hemos mencionado antes, para mitigar estos efectos introducidos por las TLBs debido a los *TLB shootdowns*, hay que implementar hardware, que se traduce en un incremento del área del SoC. En definitiva, incluso las soluciones más prácticas introducen un *trade-off* en la mejora del sistema, esto es lógico, pues nada sale gratis.

Estudios recientes, han analizado el rendimiento de las TLBs en CMPs. Bhattacharjee et al. [4] demuestra que los fallos de la TLB son predecibles y que se pueden aplicar mecanismos de cooperación y prefetching de la TLB entre cores para mejorar el rendimiento de la misma. Sin embargo, esto implica que un fallo de la TLB también debe invalidar los mapeos en los buffers de prefetch de la TLB.

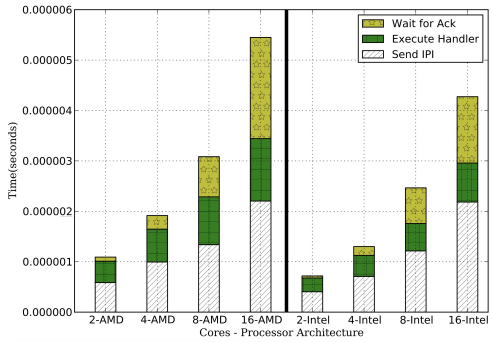

Fig. 3: Overheads de los *TLB shootdowns* con la aplicación Wordcount,Villavieja et al [5]

En la Figura 3 puede verse como el *overhead* correspondiente al envío de IPI y el asociado a la espera de los *ACKs* dependen en gran medida del número de cores. Observando

[6]Son los cores que pidieron una traducción de la entrada de tabla de páginas modificada.

el overhead de la espera de ACKs, vemos como aumenta de manera cuadrática según aumenta el $\#_{cores}$.

Es importante mencionar que las TLBs no afectan de igual manera a los SoC, esto puede verse en la Figura 3, donde en una arquitectura Intel, el tiempo es considerablemente más bajo respecto a los resultados obtenidos con una arquitectura AMD. Esto puede deberse a que la frecuencia de funcionamiento es 33.5% más alta, este incremento en la frecuencia se traduce en un decremeto en el tiempo, pues: $t \propto \frac{1}{f}$. Las especificaciones de las dos arquitecturas probadas pueden verse en la Tabla 1.

TABLE 1: ESPECIFICACIONES, VILLAVIEJA ET AL [5]

	Intel	**AMD**
Procesador	Xeon E5640	Opteron 6128
Frequencia	2.67 GHz	2 GHz
RAM	8 GB	32 GB
$\#_{cores}$	16 (4x4-chips)	16 (4x4-chips)
kernel	Linux 2.6.36	Linux 2.6.36

Si comparamos los overheads entre las dos arquitecturas, vemos como los overheads en la arquitectura Intel, son un $\sim 50\%$ más bajos.

Otra forma de entender el impacto de las TLBs en un micro, es analizando la frecuencia de los *shootdowns*, tarea ya realizada por los autores de [5]. En la Figura 4 vemos una

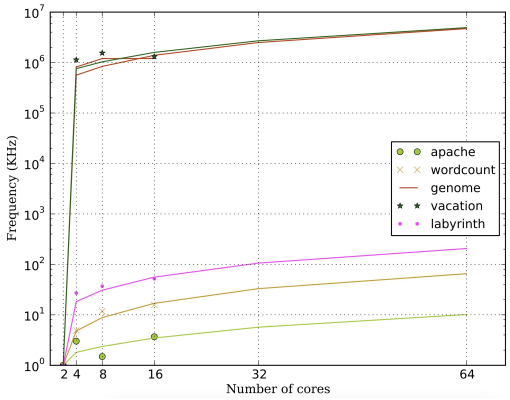

Fig. 4: Frecuencia de los *shootdowns* en función del $\#_{cores}$ en Intel, Villavieja et al [5]

proporcionalidad entre el número de cores y la frecuencia, pues según aumenta el $\#_{cores}$ la frecuencia de los *shootdowns* también lo hace, además, este incremento es lineal. Todo este análisis aplica al gráfico de la Figura 5. Donde observamos el mismo experimento, pero en una arquitectura AMD.

Como la frecuencia $F(p)$ [7] puede representarse mediante un modelo lineal, los ciclos perdidos en ejecutar *TLB shootdowns* pueden calcularse como sigue;

$$\frac{C_{shootdown} \cdot F(p)}{f_{clock}}$$

Dónde f_{clock} es la frecuencia del procesador $\therefore (f_{clock} \sim \text{GHz})$. La representación de los ciclos perdidos en la Figura 6 es de crucial importancia en temas de paralelismo, como mencioné al principio del análisis.

[7] p hace referencia al número de procesadores

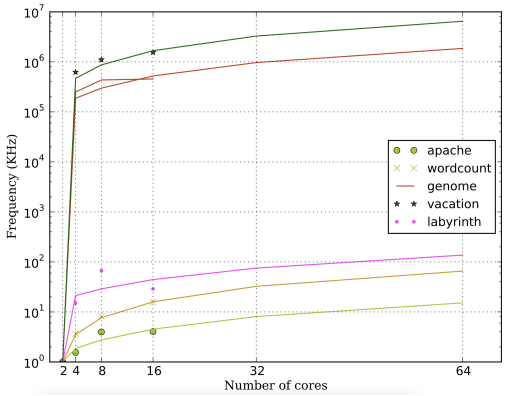

Fig. 5: Frecuencia de los *shootdowns* en función del $\#_{cores}$ en AMD, Villavieja et al [5]

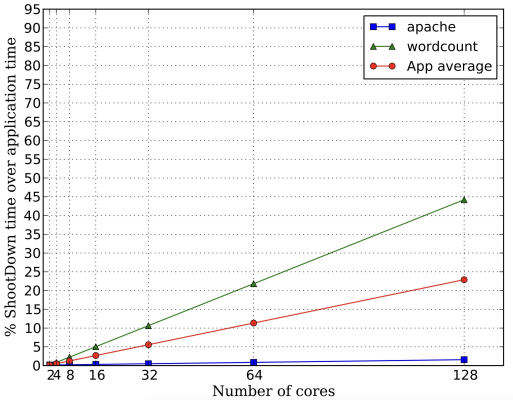

Fig. 6: Ciclos perdidos(%) ejecutando *shootdowns*, Villavieja et al [5]

Para mantener la linea del análisis, nos centramos en el (%) de ciclos perdidos ejecutando *TLB shootdowns* cuando se ejecuta la aplicación *Wordcount*. Como ya se ha visto en figuras anteriores, en la Figura 6 volvemos a ver una tendencia creciente y según aumenta el $\#_{cores}$. Analíticamente, puede verse como el (%) de ciclos perdidos ejecutando los *TLB shootdowns* aumenta en un 4% para un μ de 16 cores, y en el caso de micros de 64 y 128 núcleos el aumento lineal asciende hasta el 10% y el $\sim 24\%$ respectivamente. Esto nos muestra el gran impacto que tiene el uso de TLBs en SoCs multiprocesadores, esto es de especial importancia teniendo en cuenta el viraje hacia sistemas de multiprocesamiento de alta rendimiento.

V. IMPACTO GENERAL

Tras varias lecturas he descubierto otro articulo en el que se muestra de manera más directa el impacto de las TLBs en el renidimiento de una arquitectura. En el articulo [6] relacionan los hits y los misses con los tiempos de acceso a la memoria principal así como con los tiempos de acceso a la información contenida en la TLB.

VI. CONCLUSIONES

En definitiva, el impacto de las TLBs en los sistemas embebidos es evidente y considero que esto es de especial importancia en temas de supercomputación, donde los problemas mencionados a lo largo de este análisis están más que presentes, pues el uso de multicores es imprescindible para

obtener datos más rápidamente y resolver problemas de gran dificultad, desde el campo de la medicina hasta el origen del universo. La supercomputación, junto con la Inteligencia Artificial, el internet de las cosas y la telemedicina, conforman uno de los hitos más revolucionarios de la historia: el actual progreso tecnológico, donde las arquitecturas de computadores pueden potenciar el avance de la ciencia o convertirse en un cuello de botella. Por ello, la investigación en este área de la ingeniería es crucial para saber que tipo de TLB es más adecuado implementar en una arquitectura determinada.

En los próximos años, creo que veremos grandes avances en temas de computación cuántica y comunicaciones, hechos que serán igual o incluso más trascendentales que la Ley de Moore. Aprovecho esta reflexión, para recordar a un visionario, George Earle Moore (1929-2023), que en paz descanse.

REFERENCES

[1] A. Bhattacharjee and M. Martonosi, "Inter-core cooperative tlb for chip multiprocessors," *SIGPLAN Not.*, vol. 45, no. 3, p. 359–370, mar 2010. [Online]. Available: https://doi.org/10.1145/1735971.1736060

[2] ——, "Inter-core cooperative tlb for chip multiprocessors," in *Proceedings of the Fifteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS XV. New York, NY, USA: Association for Computing Machinery, 2010, p. 359–370. [Online]. Available: https://doi.org/10.1145/1736020.1736060

[3] B. F. Romanescu, A. R. Lebeck, D. J. Sorin, and A. Bracy, "Unified instruction/translation/data (unitd) coherence: One protocol to rule them all," *HPCA - 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, pp. 1–12, 2010.

[4] A. Bhattacharjee and M. Martonosi, "Characterizing the tlb behavior of emerging parallel workloads on chip multiprocessors," in *2009 18th International Conference on Parallel Architectures and Compilation Techniques*, 2009, pp. 29–40.

[5] C. Villavieja, V. Karakostas, L. Vilanova, Y. Etsion, A. Ramirez, A. Mendelson, N. Navarro, A. Cristal, and O. S. Unsal, "Didi: Mitigating the performance impact of tlb shootdowns using a shared tlb directory," in *2011 International Conference on Parallel Architectures and Compilation Techniques*, 2011, pp. 340–349.

[6] M. Agarwal and M. Jailia, "Effect of tlb on system performance," in *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2905055.2905280