CUP: Un Predictor de Uso de Páginas basado en Tokens

Albert Esteve, [1] Alberto Ros, [2] Antonio Robles [1] y María E. Gómez [1]

Resumen— **Distinguir la naturaleza de los datos accedidos por las aplicaciones en privados o compartidos es una propuesta que permite mejorar la gestión de la memoria a través de una gran variedad de optimizaciones para procesadores multi-núcleo. Entre los mecanismos que permiten distinguir los accesos privados, los basados en las *translation lookahead buffers* (TLBs) son una alternativa prometedora. Sin embargo, la precisión de una propuesta de clasificación basada en TLB esta estrechamente ligada con el tamaño de las TLBs. Los predictores de uso (UP –Usage Predictors) rompen esta dependencia, pero introducen una gran sobrecarga en el sistema debido al gran número de invalidaciones de TLB prematuras que introducen.**

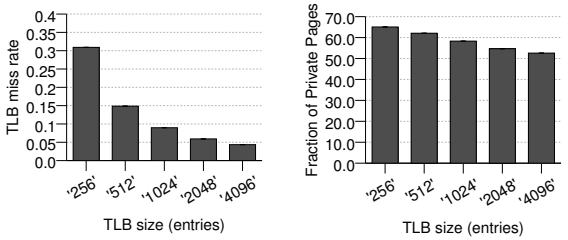
Este trabajo propone Cooperative Usage Predictor (CUP), que realiza una predicción a nivel de sistema a través de la cooperación de las TLBs. Con CUP, las invalidaciones de entradas de TLB se realizan únicamente si éstas sirven para mejorar la clasificación a privado, evitando así gran parte de la sobrecarga asociada con la predicción. CUP mejora el tiempo que una página pasa como privada (22.0% de media) sobre un mecanismo no predictivo, y evita 4 de cada 5 invalidaciones de TLB sobre otros predictores de uso. De esta forma, CUP consigue por vez primera que la predicción de uso en TLBs sea un mecanismo atractivo para la clasificación de páginas.

Palabras clave— **Data classification; Token counting; TLB; Private-shared; Read-only data; TLB Usage Predictor**

I. Introducción

Los procesadores multinúcleo (CMPs) implementan cada vez mayores estructuras de caché en un modelo de memoria compartida. Además, éstos requerirán protocolos de coherencia más escalables a medida que el número de núcleos aumenta rápidamente. Estos nuevos desafíos hacen que distinguir si los datos accedidos son privados (es decir, accedidos desde un solo hilo) o compartidos (es decir, accedidos simultáneamente desde dos o más hilos) sea una propuesta cada vez más interesante para conseguir mejorar la escalabilidad del sistema, ya que hay una gran cantidad de optimizaciones basadas en la clasificación [1], [2], [3], [4], [5].

La observación tras estas propuestas es que la mayor parte de los datos referenciados por aplicaciones tanto paralelas como secuenciales son privados. Por tanto, detectar datos privados con mayor precisión es clave para el rendimiento del sistema y los beneficios obtenidos de un mecanismo de clasificación. Por otro lado, la clasificación de datos conlleva soluciones de compromiso entre área, rendimiento y precisión. Los mecanismos basado en



(a) Tasa de fallos de la TLB según de su tamaño.

(b) Precisión de la clasificación basada en TLB según su tamaño.

Fig. 1: Análisis de la clasificación TLB y tamaño.

las TLBs [6], [7], [8] son en apariencia uno de los mejores mecanismos en términos de coste y área. Sin embargo, puesto que la clasificación es determinada por la presencia de traducciones alojadas en las TLBs, la precisión del mecanismo es dependiente del tamaño o asociatividad de las mismas. La desventaja de dicha dependencia se muestra en la Figura 1. [1] En TLBs más pequeñas, la tasa de fallo es considerablemente mayor (Figura 1a). Como es de esperar, la tasa de aciertos mejora al aumentar las entradas de la TLB de último nivel, de 256 a 4096 entradas. Sin embargo, las traducciones permaneces más tiempo en TLBs más grande, a pesar de que estas no sean accedidas de nuevo. Como consecuencia, el porcentaje de páginas privadas detectadas se reduce a medida que el tamaño de las TLBs aumenta (Figura 1b). Una estructura de TLB de tamaño ilimitado y completamente asociativa nunca fallaría más de una vez en la misma página, pero la precisión del mecanismo sería similar al de una propuesta no adaptativa (p. ej. classificación basada en el sistema operativo [3]). Idealmente, el compartimento de una página debe ser definida por la concurrencia de accesos a dicha página (es decir, concurrencia en la vida de las páginas). De ésta forma, la clasificación basada en las TLBs conseguiría una clasificación precisa, independiente de las TLBs y que no dañe la tasa de fallos de la TLB.

La predicción de uso (*Usage Prediction* –UP) [6], [7] se propuso para acercase a ésta clasificación ideal. Un predictor de uso para TLBs predice cuándo una página va a seguir siendo accedida en un core determinado. De esta forma, las traducciones son invalidadas si (i) el predictor local indica que la página correspondiente no va a seguir siendo accedida en un futuro próximo, y (ii) otra TLB solicita la traducción. De esta forma, se mejora las oportunidades de clasificar una página como privada, desacoplando

[1]Department of Computer Engineering, Universitat Politècnica de València, e-mail: alesgar@gap.upv.es, {megomez,arobles}@disca.upv.es.

[2]Departamento de Ingeniería y Tecnología de Computadores, Universidad de Murcia, e-mail: aros@ditec.um.es..

[1]Los resultados se han obtenido usando el entorno de simulación descrito en la Sección IV.

la precisión del mecanismo del tamaño de la TLB.

La observación principal de este trabajo es que las invalidaciones causadas por la predicción de uso no conllevan necesariamente una mejora en la detección de páginas privadas. De hecho, muchas invalidaciones en la TLB simplemente reducen el número de compartidores, aumentando indiscriminadamente la tasa de fallos de la TLB, sin la certeza de mejorar la clasificación de la página (Sección II). Por tanto, UP se aplica a ciegas, sin considerar cómo el estado clasificación evolucionará, a pesar de estar a su servicio. Además, una predicción poco precisa puede invalidar prematuramente entradas de TLB, lo cual aumenta dramáticamente el cantidad de fallos de TLB cuando se reaccede la página, comprometiendo así las prestaciones del sistema. Éstos factores han desalentado el uso de predictores de uso a pesar de mejorar la detección de datos privados.

Nuestra propuesta. La contribución principal de este trabajo es la de mostrar que es posible eliminar las sobrecargas de la predicción a través de un novedoso y elegante esquema de predicción basado en tokens, capaz de mantener un alto nivel de predicción de datos. Para este fin, proponemos *Cooperative Usage Predictor* (CUP), un mecanismo que explota la cooperación entre las TLBs (i.e., TLBs trabajando juntas para un propósito común) para realizar una predicción de uso a nivel de sistema (Sección III). En concreto, CUP contribuye a:

1. Realizar una predicción de uso por página basada en tokens, a nivel de sistema. De esta forma, las oportunidades de reclasificación son descubiertas sin esperar un fallo de TLB.
2. Evitar la mayor parte de las invalidaciones innecesarias de TLB, que son pospuestas hasta que éstas mejoran la clasificación, evitando así que se degraden las prestaciones (Section V) y haciendo que la predicción de uso en las TLBs sea atractiva por vez primera.

Resultados. De media, CUP pasa 22.0% más ciclos en privado que una propuesta no predictiva, haciendo además la clasificación completamente independiente del tamaño de la TLB. CUP evita 4 de cada 5 invalidaciones de TLB comparado con un predictor no cooperativo, reduciendo así la sobrecarga de la predicción. Así, cuando se aplica a la desactivación de coherencia [3], el tiempo de ejecución de las aplicaciones se mejora en un 8.8% y el consumo de energía hasta un 8.0% de media sobre otras propuestas de predicción.

II. Descripción del Problema

A. Clasificación de Datos Basada en la TLB

Los mecanismos de clasificación basadas en la TLB se apoyan en la presencia de traducciones en las TLBs del sistema (i. e., privado cuando la traducción esta presente en una TLB y compartida cuando la traducción esta presente en dos o más TLBs), siendo una de las propuestas de clasificación más precisas. Por un lado, *TLB-snooping* [7], [6] se introdujo para conseguir una clasificación apatativa en tiempo de ejecución, que descubra páginas temporalmente privadas y migración de hilos. El mecanismo de clasificación TLB-snooping se basa en comunicación de TLB a TLB para consultar las TLBs de otros núcleos del sistema tras fallos de TLB, descubriendo de forma natural si una página es actualmente privada o compartida y disminuye el tiempo de búsqueda de la página.

Por otro lado, TokenTLB [8], un mecanismo de cuenta de tokens basado en la TLB e inspirado por los protocolos de coherencia de tokens [9], fue recientemente propuesto para mitigar el tráfico TLB. TokenTLB asocia un número fijo de tokens por entrada de traducción. Usar tokens para la clasificación en lugar de usarlos para la coherencia obtiene todos los beneficios de los protocolos de tokens sin la carga de complejos mecanismos de arbitraje, ya que las carreras se resuelven como compartimento de páginas. TokenTLB mejora la precisión de la clasificación sobre anteriores esquemas de clasificación basados en TLB al favorecer una reclasificación inmediata de las páginas a privado (i. e. adaptabilidad completa), mejora el consumo de la red sobre TLB-snooping al limitar las respuestas de la TLB y la duplicación de traducciones, y completa la dicotomía de clasificación privado-compartido añadiendo la detecciones de regiones de solo lectura.

B. Predicción de Uso de la TLB

La clasificación basada en las TLBs resta en la presencia de entradas de TLB para descubrir el estado de compartimento de la página. Por otro lado, la predicción de uso permite desacoplar la clasificación de datos del tamaño de las TLBs [6]. Para este fin, la predicción de acceso a la página se realiza en cada entrada de TLB a través de un contador saturable, parecido al usado en *Cache Decay* [10]. El contador se incrementa tras un periodo fijo (es decir, el *periodo del predictor*) y reiniciado tras cada acceso de TLB a la página. Cuando el contador satura, la página cae en desuso. Tras ello, una entrada de TLB en desuso es invalidada cuando es solicitada desde la red debido a un fallo en otra TLB del sistema. De este modo, núcleos con entradas de TLB invalidadas dejan de contar como potenciales compartidores, incrementando así las posibilidades de reclasificación a privado.

Sin embargo, la reclasificación a privado solo se consigue cuando, tras un fallo de TLB, el nuevo compartidor descubre que el resto de las entradas de TLB se han desalojado o han caído en desuso. Por contra, con solo una TLB con la página aún en uso, un fallo en otra TLB resulta en una invalidación de todas las entradas en desuso sin afectar a la clasificación de ningún modo. Estas invalidaciones innecesarias son frecuentas y dañan seriamente a las prestaciones del sistema.

Emplear periodos más cortos para el predictor invalida de forma prematura más traducciones, lo cual a su vez causa que la tasa de fallos de la TLB se dis-

pare, produciendo más fallos, lo cual invalida más entradas de TLB. Es, por tanto, un problema de retroalimentación positiva (o invalidación ping-pong si solo dos TLBs están involucradas). Un predictor preciso no afectará las prestaciones, ya que los bloques de una página no serán accedidos poco después de ser invalidada.

Para evitar estas situaciones, que hacen del predictor de uso una opción poco atractiva, una modificación de la estrategia de invalidación se introdujo, conocida como *Forced Sharing* [7]. Ésta estrategia evita invalidaciones inducidas por el predictor después de que se detecte que una entrada de TLB haya sido invalidada prematuramente. En este caso, una invalidación prematura es aquella en la que el núcleo accede a la página y ésta permanece en la TLB en un estado inválido, por lo que dicho acceso se ha producido poco después de ser invalidada la entrada. Para evitar la retroalimentación, un mensaje especial *forced-sharing* es enviado, el cual sobrescribe el predictor y evita la invalidación de la entrada, aún en el caso de que este en desuso. Sin embargo, aunque ésta estrategia reduce el daño producido por un predictor agresivo, la solución depende de nuevo del tiempo que una entrada (en este caso invalidada) permanece en la TLB, y por tanto, de su tamaño.

III. Predictor de Uso Cooperativo

En este trabajo proponemos el Predictor de Uso Cooperativo (CUP), un mecanismo diseñado para trabajar conjuntamente con un mecanismo de clasificación basado en TLBs. El objetivo es el de mejorar la precisión de la predicción evitando invalidaciones en la TLB sin sentido. Para ello, las entradas de TLB reúnen información de uso de página a nivel de sistema mediante su cooperación. Dicha cooperación se consigue enviando notificaciones tan pronto como una página cae en desuso. Luego, solo si la página permanece en uso en una sola TLB, se inicia un proceso de reclasificación, invalidando las entradas de las TLBs mientras la reclasificación es posible.

A. Clasificación Basada en Tokens

CUP se implementa sobre un mecanismo de clasificación basado en tokens [8]. Éste mecanismo de clasificación basado en las TLBs es simple y preciso, asociando un número fijo de tokens por página igual al número de núcleos del sistema. La clasificación se basa en la cuenta actual de tokens en cada entrada de TLB: privada cuando tiene todos los tokens para la página; compartida cuando tiene un subconjunto de todos los tokens del sistema; e invalida cuando no tiene tokens. Los tokens se intercambian a través de mensajes entre TLBs junto con las traducciones, lo que permite una reclasificación rápida y natural a y desde privado.

Tras cada fallo de TLB se envia un solicitud a otras TLBs del sistema. Esta solicitud permite acelerar la búsqueda de traducciones en la tabla de páginas a través de la red del chip. Además, permite obtener el estado de compartimento de la página, ya que los tokens se intercambian en las respuestas de otras TLBs. La sobrecarga en la red es limitada ya que, por un lado, la tasa de fallo de las TLBs es reducida. Por otro lado, las TLBs que responden tras solicitudes de página se limitan a aquellas que contengas dos o más tokens actualmente (i. e. *token-owners*). En cada respuesta, una TLB envía uno de sus tokens, manteniendo el resto.

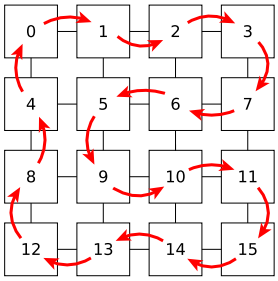


Fig. 2: Ejemplo de anillo virtual en una malla 4x4 (16 núcleos).

En una clasificación basada en tokens, éstos no puede ser generados ni destruidos. Cuando una entrada de una TLB desaloja una entrada con todos los tokens, éstos son enviados de vuelta a la tabla de páginas, donde permanecen comprimidos en un campo especial de un bit (todos los tokens o ninguno). En caso contrario, si se desaloja una entrada de TLB con un subconjunto de tokens, éstos son enviados a la red en busca de otra entrada de TLB que los retenga. Así los tokens son redistribuidos. Este proceso se conoce como desalojo no silencioso. Los tokens que recorren la red lo hacen a través de un anillo virtual como el mostrado en la Figura 2. El anillo virtual es dependiente de la topología, pero se puede implementar de forma similar para casi cualquier otra topología. Para evitar bloqueos activos, una TLB que este involucrada en un proceso de desalojo no silencioso, puede a su vez alojar tokens temporalmente.

La Figura 3 muestra el formato de las entradas TLB para realizar una clasificación basada en tokens. Los campos de gris claro representan aquellos requeridos para la clasificación específicamente: un campo de $log_2(N)$ ($tokensC$) para almacenar los tokens, un bit L para bloquear el acceso a la página cuando el estado de compartimento es incierto y un bit W para distinguir datos de solo lectura. El bit de válido (V) establece la ausencia de tokens de clasificación (es decir, la página es inválida).

B. Doble Conjunto de Tokens: Predicción

CUP se implementa de forma sencilla añadiendo un conjunto de tokens adicional asociado a cada página. De este modo, CUP emplea dos conjuntos de tokens, uno para la clasificación de páginas ($tokensC$) y el otro para predicción de uso ($tokensU$) a nivel de sistema. Así, CUP asocia cada página con un par de tokens ($\#tokensC$,$\#tokensU$), N tokens por conjunto en un sistema de N núcleos.

La Figura 3 muestra en gris oscuro los campos adicionales que se requieren con CUP en cada entrada

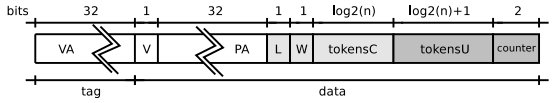


Fig. 3: Formato de entrada de TLB con los campos extra para el predictor cooperativo en gris.

de TLB. En concreto, el predictor cooperativo propuesto requiere un campo adicional de $\lceil log_2(N+1) \rceil$ bits en comparación con predictores de uso anteriores. Éste campo indica cuántas TLBs están usando actualmente la página en todo el sistema. CUP también require un campo contador de 2 bits, que es el que indica cuándo una página cae en desuso (tras cuatro *timeouts*). Éste campo es usado por todos los predictores de uso en TLBs. Así que en total, nuestra estrategia de doble conjunto requiere en total $3 + \lceil log_2(N+1) \rceil * 2$ bits, lo cual supone una sobrecarga de un ∼15% o un ∼25% del área de la L2 TLB para sistemas de 16 o 1024 núcleos respectivamente, según CACTI [11].

La presencia de *tokensU* almacenados en una entrada de TLB suponen que el núcleo es un potencial compartidor de la página. Cuando dicha entrada cae en desuso, los *tokensU* son enviados a otro núcleo distinto que esté usando la página. Aún así, los accesos a dicha página son resueltos normalmente aún tras la ausencia de *tokensU*, mientras siga siendo válida (al menos 1 *tokenC*). Los *tokensU* tampoco son necesarios para responder a un fallo de otra TLB, mientras tenga más de un *tokenC*. Aún así, si también contiene más de un *tokenU*, ésta responde con (1,1) tokens a la TLB que falló.

De forma similar, en los desalojos no silenciosos, un mensaje que contenga algún *tokenU* solo puede ser recogido por una TLB que este en uso. De esta forma se asegura que si una TLB recupera todos los *tokensU* del sistema, probablemente sea la única TLB actualmente en uso, descubriendo una oportunidad de reclasificación a privado si ésta no posee todos los *tokensC*.

B.1 Obsolescencia de entradas de TLB

CUP evita la invalidación de entradas de TLB hasta que se descubre una oportunidad de reclasificación. La cooperación se basa en anunciar la condición de desuso cuando ocurre, para tener la cuenta de TLBs que usan la página actualizada.

Principio #1: mantener la página privada mientras sea posible. Mientras una página es privada (contiene los (N, N) tokens) solo una TLB mantiene su traducción, por tanto al entrar en desuso no se envía ningún mensaje anunciándolo. Si otra TLB falla y se encuentra la TLB en desuso, Todos los tokens son enviados a la red, invalidando la entrada y favoreciendo que la página permanezca privada.

Principio #2: cooperar para detectar oportunidad de reclasificación. Cuando una entrada de TLB cae en desuso en una página compartida, ésta envía todos sus tokens menos uno $(\#tokensC - 1, \#tokensU)$ a la red, siguiente el anillo virtual de la Figura 2. Éste proceso se conoce por *notificación de desuso*.

Sólo una TLB que tenga una entrada válida y en uso puede recoger tokens de una notificación de desuso. A diferencia de los mensaje de desalojo, no es necesario esperar un ACK. Así, durante el proceso, se permite el acceso a la página como compartida. En caso de que el mensaje vuelva a su origen, éste recoge sus propios tokens y desactiva la notificación. De esta forma, una entrada de una página compartida que este en desuso puede, aún así, contener *tokensU*. Los tokens deberán ser enviados $(\#tokensC - 1, \#tokensU)$ tras un fallo en otra TLB si ésta permanece en desuso.

B.2 Reclasificación *Cruise-missile*

En un momento dado, una entrada de TLB de una página compartida recuperará los N *tokensU*, sugiriendo que puede ser la única TLB que está accediendo la página, mientras otras retienen aún sus *tokensC*. Así pues, se inicia un proceso de reclasificación para poder recuperar los *tokensC*. Para ello, empleamos una estrategía llamada *cruise-missile reclassification* (CMR), que tiene alguna similaridad con *cruise-missile-invalidates* [12]. El proceso de reclasificación envía un mensaje al anillo virtual. El mensaje CMR mantiene un par $(\#tokensC, \#tokensU)$, conocida como *token pool*, inicializado a $(0, N - \#tokensC)$. El primer componente recopila los *tokensC* recuperados de otras TLBs, mientras que el segundo componente representa la cantidad de *tokensC* que deben ser recuperados. Los *tokensU* in la *token pool* del CMR permiten restaurar una entrada en que ha sido reaccedida. El objetivo del mensaje CMR es recorrer el anillo virtual hasta que los *tokensC* se recuperen completamente, invalidando el resto de entradas de TLB.

Por otro lado, los accesos a memoria en el núcleo que inicia el proceso de reclasificación no se bloquean. La clasificación evoluciona de forma natural cuando los tokens atraviesan el anillo virtual. Así se evita el impacto negativo en el rendimiento del sistema. El CMR retrasa la reclasificación, causando temporalmente algunos accesos compartidos a datos posiblemente privados.

Por último, la reclasificación no requiere que se atraviese el anillo virtual por completo, permitiendo algunas optimizaciones para evitarlo.

Caso #1: reclasificación exitosa. En cada salto de un mensaje CMR, si la traducción esta presente y en desuso, ésta es invalidada, desalojando los bloques en la L1 y almacenando los *tokensC* en la *token pool*. El mensaje CMR sigue recuperando tokens hasta que $\#tokensC = \#tokensU$, lo que hace que el proceso de reclasificación termine exitosamente, y el mensaje vuelve al núcleo que inició el proceso.

Caso #2: restaurar la página desde el *token pool*. Los *tokensU* en la *token pool* del CMR permiten restaurar una traducción, haciendo que cuente de nuevo como compartidor para dicha página. Cuando una TLB recibe un mensaje CMR teniendo una traducción que ha sido accedida recien-

TABLA I: Parámetros del sistema base.

Parámetros de memoria			
Frecuencia del procesador	2.8GHz	Jerarquía TLB	Exclusive
L1 TLBs separadas instr & datos	8 conjuntos, 4 vías	L1 TLB tiempo de acierto	1 ciclo
L2 TLB combinada	128 conjuntos, 4 vías	L2 TLB tiempo de acierto	2 ciclos
Timeout adquisición de tokens	1200 ciclos	TPB	32 conjuntos, 4 vías
TPB tiempo de acierto	1 ciclo	Tamaño de página	4KB (64 bloques)
Jerarquía caché	No inclusiva	Tamaño de bloque caché	64 bytes
L1 cachés separadas instr & data	64KB, 4 vías	L2 cache combinada compartida	1MB/tile, 8 vías
L1 cache tiempo de acierto	1 (tag) y 2 (tag+datos) ciclos	L2 cache tiempo de acierto	2 (tag) y 6 (tag+datos) ciclos
Caché de directorio	256 sets, 4 ways	Directorio tiempo de acierto	1 ciclo
Tiempo de acceso a memoria	160 ciclos		
Parámetros de red			
Topología	Malla 2D (4x4)	Mecanismo de encaminamiento	Determinístico X-Y
Tamaño de flit	16 bytes		
Tiempo de encaminamiento, switch y enlace	2, 2 y 2 ciclos	Tamaño de mensajes de datos y control	5 flits y 1 flit

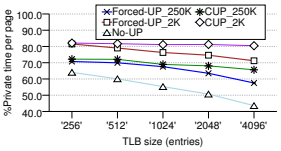
temente, ésta reclama tantos *tokensU* como *tokensC* mantiene, restaurando la entrada y abortando la reclasificación. El mensaje es enviado de vuelta, ya que la reclasificación no se puede completar.

Caso #3: fallo de TLB durante una reclasificación. Por último, una reclasificación a privado se cancela cuando un mensaje CMR cruza una TLB que o bien ha obtenido tokens (por ejemplo, tras resolver un fallo de TLB para esa página), o bien esta actualmente envuelta en un fallo de TLB. Puesto que la reclasificación empieza cuando todos los *tokensU* están en una sola TLB, la presencia de tokens en otra TLB distinta a la que inició el proceso, implica que han sido obtenido recientemente, impidiendo que se complete la reclasificación.

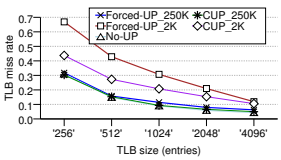
IV. Metodología de Evaluación

La propuesta es evaluada mediante simulaciones de sistema completo, con Virtutech Simics [13] y Wisconsin GEMS [14], que permite una simulación detallada de sistemas multinúcleo. La red de interconexión se ha modelado usando el simulador GARNET [15]. La arquitectura base se trata de un CMP de 16 nodos con un sistema de coherencia cache de directorio, cuyos parámetros son los mostrados en la Tabla I. La latencia y el consumo de las cachés y las TLBs se han calculado usando la herramienta CACTI [11], asumiendo una tecnología de 32nm. Los periodos del predictor evaluados se basan en el estudio en [7]: 250K, 50K, 10K, y 2K ciclos.

La evaluación de la propuesta esta realizada a través de una amplia variedad de cargas de trabajo y aplicaciones paralelas de varias suites, cubriendo así distintos patrones y grados de compartición: Barnes *(8192 bodies, 4 time steps),* Cholesky *(tk15.O),* FFT *(64K complex doubles),* Ocean *(258 × 258 ocean),* Radiosity *(room, -ae 5000.0 -en 0.050 -bf 0.10),* Raytrace-opt *(teapot, optimized by removing a lock acquisition for a ray ID which is not used),* Volrend *(head), and* Water-NSQ *(512 mol., 4 time steps) are from the SPLASH-2 benchmark suite [16].* Tomcatv *(256 points, 5 time steps) and* Unstructured *(Mesh.2K, 5 time steps) are two scientific benchmarks.* FaceRec *(script) and* SpeechRec *(script) belong to the ALPBenchs suite [17].* Blackscholes *(simmedium),* Swaptions *(simmedium), and* x264 *(simsmall) come from PARSEC [18]. Finally,*



(a) Proporción de tiempo en Privado.



(b) Tasa de fallos de TLB.

Fig. 4: Tamaño de TLB y análisis de predicción.

Apache *(1000 HTTP transactions) and* SPEC-JBB *(1600 transactions) are two commercial workloads [19].* Todos los resultados corresponden a la fase paralela de dichas aplicaciones.

V. Resultados

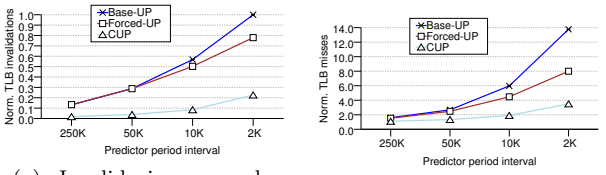
En esta sección comparamos CUP con otros predictores y distintos periodos de predicción, y su eficiencia aplicado un caso de uso, como es la desactivación de la coherencia.

A. Predicción de Uso

Influencia del tamaño de la TLB. El propósito principal al emplear un predictor de uso es el de conseguir que la precisión de la clasificación sea independiente del tamaño de la TLB en mecanismos basado en la TLB. Para ilustrar esto, la Figura 4 muestra cómo la clasificación varía al variar el tamaño de la TLB (con la misma asociatividad).

Por un lado, la proporción de ciclos que una página permanece como privada de media se muestra en la Figura 4a. Como es de esperar, *No-UP* disminuye gradualmente la proporción de tiempo privado por página a medida que el tamaño de la TLB aumenta. De forma similar, *Forced-UP* es afectado por las variaciones en el tamaño de la TLB, perdiendo precisión progresivamente. Por contra, *CUP* no varía, manteniendo un 22% más página en privado de media con *CUP_250K*, y hasta un 36.8% con *CUP_2K*, con respecto a *No-UP*. Además, *CUP_250K* permanece 8.1% más ciclos como privado que *UP_250K*.

Por otro lado, la Figura 4b muestra como la tasa de fallos de la TLB evoluciona con los distintos tamaños de TLB. Aplicar las técnicas de predicción conlleva un coste, ya que implica aumentar la tasa de fallos al reducir el periodo del predictor. Además, cuanto mayor es la TLB, menor es su tasa de fallos. Se puede observar como la tasa de fallos evoluciona más



(a) Invalidaciones en la TLB inducidas por el predictor.

(b) Fallos de TLB normalizados.

Fig. 5: Análisis de predicción.

rápidamente con *Forced-UP*, ya que muchas invalidaciones causadas por retroalimentación positiva se evitan de forma natural al incrementar el tamaño de la TLB. Sin embargo, reducir la tasa de fallo de la TLB reduce la precisión de esquema de clasificación. Por contra, la pérdida de precisión se previene a través de la cooperación de las TLBs, como se muestra en la Figura 4a, ya que las oportunidades de reclasificación son descubiertas sin esperar fallos en la TLB.

Análisis de sobrecostes de la predicción. Una de las principales contribuciones de CUP es la de invalidar las entradas de TLB solo cuando una página puede potencialmente transitar a privado, lo cual reduce dramáticamente la cantidad de invalidaciones inducidas por el mecanismo de predicción, así como sus fatales consecuencias. En la Figura 5a, normalizada a *Base-UP*, se muestra como *Forced-UP* disminuye la cantidad de invalidaciones inducidas por el predictor hasta un 22% con un predictor de 2K. De forma distinta, CUP mantiene casi invariable la proporción de invalidaciones adicionales en todas las configuraciones, aunque en el periodo más agresivo, aún se inducen algunas invalidaciones adicionales. Dichas invalidaciones degradan la precisión del predictor, favoreciendo una mejor detección de privados. Aún así, se evitan 4 de cada 5 invalidaciones en comparación con *Base-UP*.

Therefore, eliminating most miss-predicted invalidations in the TLB implies reducing the amount of predictor-induced TLB misses, as shown in Figure 5b. TLB misses in the figure are normalized to a non-predicting token-counting classification. Specifically, *CUP* halves the amount of TLB misses compared to *Forced-UP*, particularly on greater predictor periods.

B. Desactivación de la Coherencia

La desactivación de la coherencia es la optimización escogida como caso de estudio para estudiar el comportamiento del esquema de clasificación propuesto.

La Figura 6 representa el tiempo de ejecución para mecanismos de clasificación basados en tokens, con y sin predicción. Todos los resultados están normalizados a un sistema sin desactivación de la coherencia o transferencias entre TLBs. Los periodos de 10K y 2K no se muestran, por mejorar la claridad de los datos. A pesar de que éstos reducen el uso del directorio en mayor grado, tanto en cuanto aumentan la tasa de fallos de las cachés o la TLB en 3.6% y 2.0% respectivamente, su uso puede no ser indicado. Como

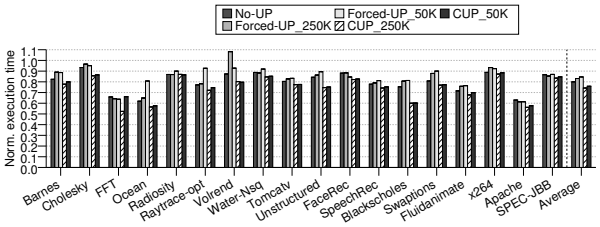


Fig. 6: Tiempo de ejecución.

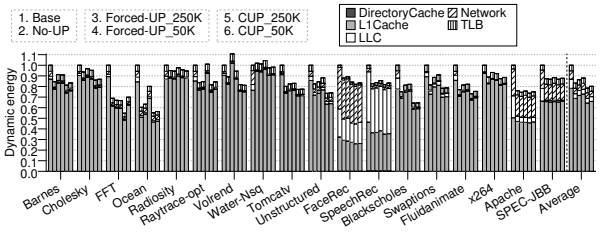


Fig. 7: Consumo dinámico de energía en la jerarquía caché.

se puede observar, desactivar la coherencia mejora el tiempo de ejecución en un 20% con *No-UP*. *Forced-UP* no consigue superar sus sobrecostes con los beneficios que se obtienen de una mejor clasificación al aplicar la desactivación de la coherencia. De hecho, la invalidación masiva de entradas de TLB daña las prestaciones respecto a *No-UP*, en un 3.1% y un 4.7% respectivamente, con periodos de 250K y 50K respectivamente. Por el contrario, CUP saca todo el provecho de la desactivación de la coherencia, eliminando los sobrecostes de la predicción a través de predicciones precisas. Con todo, CUP tiene un impacto positivo en el tiempo de ejecución, no solo evitando la pérdida de prestaciones, sino mejorándolas con *CUP_250K* en un 5.8% comparado con *No-UP*, 8.8% sobre *Forced-UP_250K*, y hasta un 25.8% sobre el sistema base.

Además, mejorar la precisión de la clasificación y reducir los sobrecostes tiene también un impacto directo en el consumo de la jerarquía caché. La Figura 7 muestra como el consumo se reduce en un 16.9% con *No-UP* en comparación con *Base*, ya que desactivar la coherencia reduce la tasa de fallos en la caché L1. Sin embargo, *Forced-UP* aumenta el consumo sobre *No-UP* en 3.3% y 4.8% para periodos de predicción de 250K y 50K, respectivamente, a causa del daño de los fallos inducidos en la TLB y la caché. Por contra, *CUP_250K* reduce el consumo dinámico en un 4.7% sobre *No-UP* y 8.0% con respecto a *Forced-UP*.

VI. Conclusiones

En este trabajo proponemos un Predictor de Uso Cooperativo (CUP) para TLBs, un mecanismo diseñado para reducir el tiempo de generación de las traducciones de página en las TLBs de acuerdo a su uso, para así servir una clasificación más precisa. CUP es independiente del tamaño de las TLBs, revela de forma inmediata las transiciones a privado e invalida las entradas de TLB solo cuando se beneficia una reclasificación a privado. Para ello, introducimos una estrategia de dos conjuntos de tokens.

Uno de los conjuntos se usa para la clasificación de páginas, mientras que el otro representa el uso de la página en las TLBs del sistema. Anunciando los periodos de desuso en mensajes en los que los tokens de uso son enviados a la red, la oportunidad de reclasificación es revelada de forma natural. Entonces, nuestra estrategia *cruise-missile* permite que el proceso de reclasificación tenga un bajo coste en término de prestaciones o tráfico. Como consecuencia, CUP permite optimizar los accesos privados con un sobrecoste mínimo, lo que hace que la predicción sea una estrategia atractiva por primera vez, promoviendo una mejor clasificación y mejorando las prestaciones.

Agradecimientos

Este trabajo ha sido financiado por MINECO y la Comisión Europea (fondos FEDER) bajo el proyecto TIN2015-66972-C5-1-R/3-R y la *Fundación Séneca, Agencia de Ciencia y Tecnología de la Región de Murcia* bajo el proyecto *Jóvenes Líderes en Investigación* 18956/JLI/13.

Referencias

[1] Nikos Hardavellas, Michael Ferdman, Babak Falsafi, and Anastasia Ailamaki, "Reactive NUCA: Near-optimal block placement and replication in distributed caches," in *36th Int'l Symp. on Computer Architecture (ISCA)*, June 2009, pp. 184–195.

[2] Daehoon Kim, Jeongseob Ahn, Jaehong Kim, and Jaehyuk Huh, "Subspace snooping: Filtering snoops with operating system support," in *19th Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT)*, Sept. 2010, pp. 111–122.

[3] Blas Cuesta, Alberto Ros, María E. Gómez, Antonio Robles, and José Duato, "Increasing the effectiveness of directory caches by deactivating coherence for private memory blocks," in *38th Int'l Symp. on Computer Architecture (ISCA)*, June 2011, pp. 93–103.

[4] Abhayendra Singh, Satish Narayanasamy, Daniel Marino, Todd Millstein, and Madanlal Musuvathi, "End-to-end sequential consistency," in *39th Int'l Symp. on Computer Architecture (ISCA)*, June 2012, pp. 524–535.

[5] Alberto Ros and Stefanos Kaxiras, "Complexity-effective multicore coherence," in *21st Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT)*, Sept. 2012, pp. 241–252.

[6] Alberto Ros, Blas Cuesta, María E. Gómez, Antonio Robles, and José Duato, "Temporal-aware mechanism to detect private data in chip multiprocessors," in *42nd Int'l Conf. on Parallel Processing (ICPP)*, Oct. 2013, pp. 562–571.

[7] Albert Esteve, Alberto Ros, María E. Gómez, Antonio Robles, and José Duato, "Efficient tlb-based detection of private pages in chip multiprocessors," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 27, no. 3, pp. 748–761, Mar. 2016.

[8] Albert Esteve, Alberto Ros, Antonio Robles, María E. Gómez, and José Duato, "Tokentlb: A token-based page classification approach," in *Int'l Conf. on Supercomputing (ICS)*, June 2016.

[9] Milo M.K. Martin, *Token Coherence*, Ph.D. thesis, University of Wisconsin-Madison, Dec. 2003.

[10] Stefanos Kaxiras and Georgios Keramidas, "SARC coherence: Scaling directory cache coherence in performance and power," *IEEE Micro*, vol. 30, no. 5, pp. 54–65, Sept. 2011.

[11] Shyamkumar Thoziyoor, Naveen Muralimanohar, Jung Ho Ahn, and Norman P. Jouppi, "Cacti 5.1," Tech. Rep. HPL-2008-20, HP Labs, Apr. 2008.

[12] Luiz A. Barroso, Kourosh Gharachorloo, Robert McNamara, Andreas Nowatzyk, Shaz Qadeer, Barton Sano, Scott Smith, Robert Stets, and Ben Verghese, "Piranha: A scalable architecture based on single-chip multiprocessing," in *27th Int'l Symp. on Computer Architecture (ISCA)*, June 2000, pp. 12–14.

[13] Peter S. Magnusson, Magnus Christensson, Jesper Eskilson, Daniel Forsgren, Gustav Hallberg, Johan Hogberg, Fredrik Larsson, Andreas Moestedt, and Bengt Werner, "Simics: A full system simulation platform," *IEEE Computer*, vol. 35, no. 2, pp. 50–58, Feb. 2002.

[14] Milo M.K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood, "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset," *Computer Architecture News*, vol. 33, no. 4, pp. 92–99, Sept. 2005.

[15] Niket Agarwal, Tushar Krishna, Li-Shiuan Peh, and Niraj K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator," in *IEEE Int'l Symp. on Performance Analysis of Systems and Software (ISPASS)*, Apr. 2009, pp. 33–42.

[16] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," in *22nd Int'l Symp. on Computer Architecture (ISCA)*, June 1995, pp. 24–36.

[17] Man-Lap Li, Ruchira Sasanka, Sarita V. Adve, Yen-Kuang Chen, and Eric Debes, "The ALPBench benchmark suite for complex multimedia applications," in *Int'l Symp. on Workload Characterization (IISWC)*, Oct. 2005, pp. 34–45.

[18] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *17th Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT)*, Oct. 2008, pp. 72–81.

[19] Alaa R. Alameldeen, Carl J. Mauer, Min Xu, Pacia J. Harper, Milo M.K. Martin, Daniel J. Sorin, Mark D. Hill, and David A. Wood, "Evaluating non-deterministic multi-threaded commercial workloads," in *5th Workshop On Computer Architecture Evaluation using Commercial Workloads (CAECW)*, Feb. 2002, pp. 30–38.