Apéndice A

Análisis de Radix sort

El objetivo en este apéndice es obtener un valor para el tamaño del dígito a usar con Radix sort. Como se ha dicho en el Capítulo 2, Radix sort se usa cuando los datos a ordenar caben en un cierto nivel de la jerarquía de memoria y además no producen un número significativo de fallos de TLB. Así pues, dadas estas circunstancias se ha construido un simulador que permite calcular el tamaño de los dígitos para Radix sort en distintas situaciones suponiendo que los datos caben en el segundo nivel de memoria cache.

Análisis del número de dígitos óptimo

Suponemos un modelo para CSE donde: $CSE = CSE_{mem} + CSE_{cpu}$. CSE_{mem} se calcula con el simulador y es el número total de ciclos de penalización por fallos en la jerarquía de memoria dividido por el número total de elementos ordenados. El número de CSE_{cpu} se ha calculado según el modelo de ejecución en orden para el MIPS R10K. El primer nivel de cache simulado es de mapeo directo con capacidades de $8K$, $16K$, $32K$ y $64K$ bytes. El tamaño de la línea de cache de este nivel es de 32 bytes. Sin embargo, se ha supuesto que los datos y las estructuras necesarias para la ordenación están y caben en el segundo nivel de cache, por lo que no se producirán fallos accediendo a este nivel. Los tamaños de los conjuntos de datos son $2K$, $16K$ y $128K$ datos. Cada dato tiene una clave de 32 bits y un puntero de 32 bits. $128K$ es el máximo número de tuplas que caben en el segundo nivel de cache, fuera del chip, de un R10K sin tener un número de fallos significativo de TLB al escribir en el vector destino. Se ha utilizado la penalización de fallo del primer nivel de cache (dentro del chip) de un R10K, que es de 9 ciclos.

En este apéndice suponemos que Radix sort tiene que ordenar un subconjunto de datos que pueden haber sido particionados anteriormente. En este caso, podemos encontrarnos con claves ordenadas parcialmente, es decir, que sólo tienen b bits no

ordenados. Se han realizado simulaciones para $b = 31$, $b = 26$, $b = 21$ y $b = 16$ bits, lo cual nos permiten ver un espectro amplio de casos, y se han dividido los b bits de la clave en 1 a 5 dígitos para obtener los valores óptimos en los conjuntos de datos que se han considerado. Por ejemplo, para el caso de $b = 26$, se han considerado los casos de 2 dígitos de 13 bits cada uno, 3 dígitos con 8 bits el primer dígito y 9 bits el segundo y tercero, 4 dígitos con dígitos de 6, 6, 7 y 7 bits respectivamente, 5 dígitos con dígitos de 5, 5, 5, 5 y 6 bits.

Resultados de la simulación

El análisis se realiza para la ordenación de un conjunto de datos que tiene una distribución Random donde las claves tienen un tamaño de 32 bits. De esos 32 bits sólo se ordenarán los b bits de menor peso, suponiendo que los de más peso son iguales.

Aquí se muestra un ejemplo del análisis para $b = 26$ bits. Se empieza analizando los ciclos invertidos en el acceso a la jerarquía de memoria y se sigue con el análisis de los ciclos empleados sólo en el procesador, cuando no hay fallos de cache. Finalmente, se analiza el número total de ciclos de la ordenación.

Análisis del subsistema de memoria

La figura A.1 muestra la simulación de los CSE_{mem} para ordenar claves y punteros de 32 bits donde sólo faltan por ordenar $b = 26$ bits de la clave. Las diferentes curvas corresponden a tamaños de primer nivel de memoria cache de 8K, 16K, 32K y 64K bytes.

En la Figura se observa que cuando el número de dígitos es pequeño (2 en oposición a 3, 4 ó 5) los CSE_{mem} aumentan considerablemente. Con el simulador se distinguen los fallos en el acceso a las diferentes estructuras de datos, viéndose cómo interfieren las unas con las otras. Se comprobó que este aumento es debido a que las interferencias entre contadores y entre contadores y los vectores S y D aumentaban. Los CSE_{mem} disminuyen cuando se utilizan caches grandes ya que el número de interferencias es menor.

Cuando el número de dígitos es grande (5 en oposición a 2, 3 ó 4) los fallos de cache también aumentan. Eso ocurre si los vectores S y D no caben en el primer nivel de memoria cache (16K y 128K). Este crecimiento se debe a que el número de pasos de movimiento aumenta linealmente con el número de dígitos. Por lo que, si los dos vectores no caben en el primer nivel de cache, cuando se empieza otro movimiento, se deben buscar en el segundo nivel de cache los datos que no se pudieron mantener en el primer nivel. Sin embargo, en el caso de que los vectores S y D quepan en el primer nivel de cache, el número de fallos no crece. En la Figura A.1 se puede observar que para $2K$ claves y punteros y caches de 32K y 64K no aumenta el número de fallos

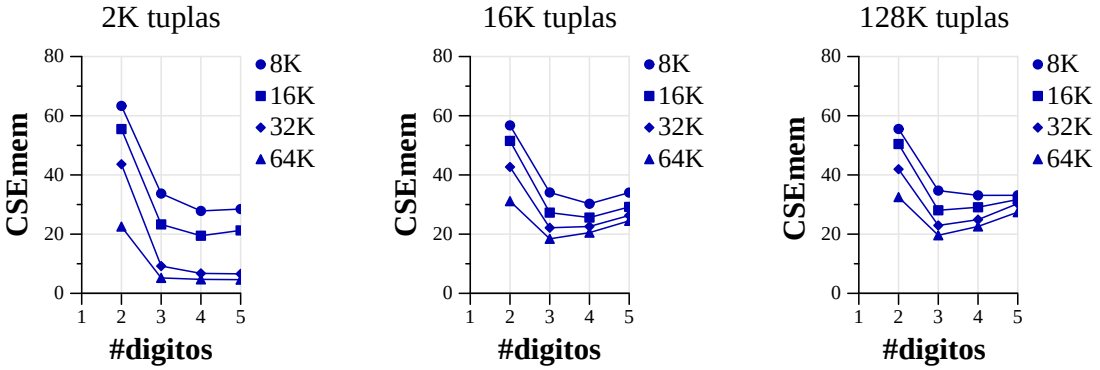

Figura A.1: CSE_{mem} para la ordenación con Radix sort de claves con $b = 26$ bits a ordenar y una distribución Random.

al acceder al primer nivel de cache, a medida que aumentamos el número de dígitos. Esto se debe a que los vectores S y D caben en ambas caches.

Análisis del procesador

En la Figura A.2 se muestran los CSE_{cpu} que se han obtenido para el algoritmo en un R10K utilizando los contadores hardware de este procesador.

Se sabe que Radix sort debe realizar tantos pasos de movimiento como números de dígitos tenga la clave. Si el número de dígitos aumenta, los CSE_{cpu} también crecen para conjuntos de datos grandes, ya que el número de pasos de movimiento también se incrementa. Esto se ve con claridad en la Figura A.2.

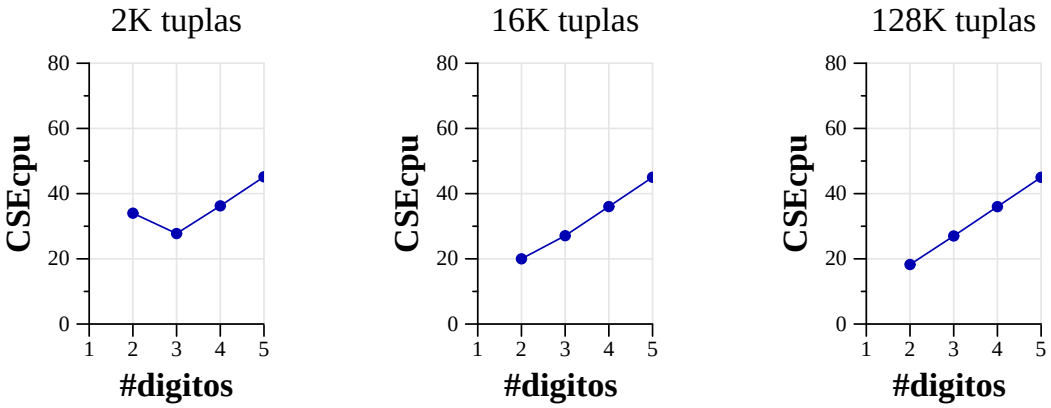

Figura A.2: CSE_{cpu} para la ordenación con Radix sort de claves con $b = 26$ bits a ordenar y una distribución Random.

Además, para conjuntos de datos pequeños (2K tuplas), el número de CSE_{cpu} para

2 dígitos es mayor que para 3 dígitos. Esto se debe a que el número de ciclos invertidos en los pasos de inicialización y acumulación es considerable cuando el número de contadores necesarios para un dígito es mayor que el número de claves del conjunto de datos. Sin embargo, el aumento del número de CSE_{cpu} es menor que el que se tenía en CSE_{mem}.

Análisis de los CSE totales

En la Figura A.3 se muestran los CSE totales obtenidos en la simulación para diferentes tamaños del primer nivel de memoria cache. Los CSE totales se calculan sumando los CSE_{mem} y los CSE_{cpu}. Se puede observar que el número de CSE totales mínimo se obtiene para 3 dígitos, sea cual sea el número de tuplas y el tamaño de cache.

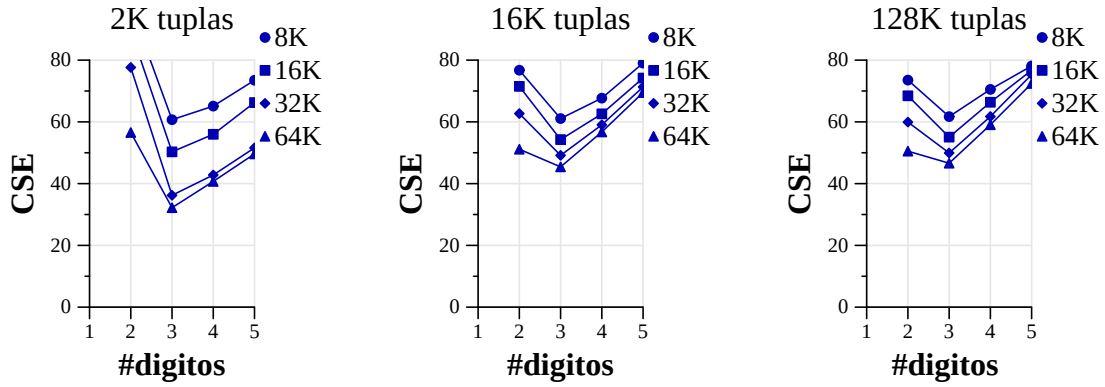

Figura A.3: CSE totales para la ordenación con Radix sort de claves con $b = 26$ bits a ordenar y una distribución Random.

Conclusión

De este estudio se puede concluir que para obtener los mejores resultados se debe usar el **mínimo número de dígitos de tamaño similar que hace que todos los contadores quepan, a la vez, en el primer nivel de memoria cache**. En la Tabla A.1 se muestra el número óptimo de dígitos cuando el tamaño del primer nivel de memoria cache va de $8K$ a $64K$ bytes y b varía entre 16 y 31 bits para ordenar $128K$ datos de clave y puntero.

Tamaño de Cache	8K	16K	32K	64K
$b = 31$	4/3.5K	3/16K	3/16K	3/16K
$b = 26$	3/5K	3/5K	3/5K	3/5K
$b = 21$	3/1.5K	2/12K	2/12K	2/12K
$b = 16$	2/2K	2/2K	2/2K	2/2K

Tabla A.1: Número óptimo de dígitos y tamaño de memoria ocupada por los contadores de todos los dígitos separados por el símbolo /. Los resultados son para un primer nivel de memoria cache de $8K$ a $64K$ bytes y para un número de bits b a ordenar de la clave para ordenar 128K datos de clave y puntero (1K = 1024). Los contadores son de 4 bytes cada uno.