Informe Tarea 4
Arquitectura de Computadores

Rafael Fernández

1 Desarrollo

Para el presente informe se desarrolló un *software* capaz de simular los accesos a memoria de una serie de programas entregados en formato *.json*. El software simula los estados de la memoria física, la caché, la TLB y la tabla de páginas de cada programa en cada acceso a memoria. Los parámetros de cada uno de los componentes mencionados anteriormente, correspondientes a políticas de reemplazo, tamaños en *bytes*, entre otros, se entregan como *inputs* al programa. En base a dicha información, el programa genera una serie de estadísticas por programa. Estas serán estudiadas a continuación para determinar de manera empírica cuales son los parámetros que generan un mejor rendimiento.

Se empezará por estudiar el comportamiento del conjunto de programas del archivo *test1.json*. Las estadisticas resultantes de la simulación con los parámetros por defecto del archivo fueron los siguientes:

Programa	**HRT**	**HRC**	**PF**	**SO**	**SI**	**ST**	**TT**	**CT**	**MT**
Paint	0.73	0.64	1	0	0	0	2230	1352	2150
Chrome	0.8	0.6	1	0	0	0	2020	1232	2150
Telegram	0.8	0.4	1	0	0	0	1010	624	1650
Sublime	0.86	0.43	1	0	0	0	1410	872	2150

Para empezar, cambiaremos la política de reemplazo de la TLB de LFU a LRU.

Programa	**HRT**	**HRC**	**PF**	**SO**	**SI**	**ST**	**TT**	**CT**	**MT**
Paint	0.73	0.64	1	0	0	0	2230	1352	2150
Chrome	0.8	0.6	1	0	0	0	2020	1232	2150
Telegram	0.8	0.4	1	0	0	0	1010	624	1650
Sublime	0.86	0.43	1	0	0	0	1410	872	2150

Como podemos ver, no hubo cambios respecto a la versión original. Esto puede explicarse por el hecho de que los accesos a memoria virtual son direcciones que corresponden a la página cero de la memoria virtual (Ver hit rate alto de la TLB), por lo que no es necesario cambiar reemplazar páginas en la TLB ya que esta nunca se llena, provocando que los cambios en la política de reemplazo no afecten el rendimiento.

Si ahora cambiamos el valor de la función de correspondencia de la caché de *fully associative* a *2-way-associative* obtenemos:

Programa	HRT	HRC	PF	SO	SI	ST	TT	CT	MT
Paint	0.73	0.64	1	0	0	0	2230	1352	2150
Chrome	0.8	0.7	1	0	0	0	2020	1224	1650
Telegram	0.8	0.4	1	0	0	0	1010	624	1650
Sublime	0.86	0.43	1	0	0	0	1410	872	2150

Como se puede ver, el Hit Rate de la caché al usar Chrome mejoró de 0.6 a 0.7, permitiendo así que el tiempo total de memoria disminuyera en 500 nanosegundos, significando una mejora de un 23% menos de tiempo.

Ya que en este caso tenemos una caché grande en comparación a los accesos totales de memoria (128 bytes vs 34 accesos), se procederá a reducir el tamaño de la caché a 8 bytes para poder apreciar como afectan realmente las políticas de reemplazo y funciones de correspondencia en la memoría caché. Los resultados luego del cambio fueron los siguientes:

Programa	HRT	HRC	PF	SO	SI	ST	TT	CT	MT
Paint	0.73	0.27	1	0	0	0	2230	1384	4150
Chrome	0.8	0.5	1	0	0	0	2020	1240	2650
Telegram	0.8	0.4	1	0	0	0	1010	624	1650
Sublime	0.86	0.43	1	0	0	0	1410	872	2150

Ahora que el espacio en la caché es escaso, los reemplazos se empiezan a hacer más comunes y el tiempo de una entrada en la caché es menor. Esto se ve reflejado en las bajas de los hit rates de caché, donde destaca la asociada al programa Paint, quien disminuyó su hit rate de 64% a 27%.

A continuación se probará cambiando la política de reemplazo de la caché de LRU a LFU. Se obtuvieron los siguientes resultados.

Programa	HRT	HRC	PF	SO	SI	ST	TT	CT	MT
Paint	0.73	0.27	1	0	0	0	2230	1384	4150
Chrome	0.8	0.5	1	0	0	0	2020	1240	2650
Telegram	0.8	0.4	1	0	0	0	1010	624	1650
Sublime	0.86	0.29	1	0	0	0	1410	880	2650

Se observa que disminuyó el hit rate de los accesos a la caché del programa Sublime, mientras los otros programas se mantuvieron iguales. Esto se puede asociar a que los accesos a memoria del programa Sublime no son tan repetitivos (como se podría dar en el caso de un ciclo), si no más bien que hace referencia a direcciones recientemente usadas, como podría serlo un largo procedimiento donde se van definiendo variables auxiliares para cada etapa. Se concluye que conviene LRU para este tipo de accesos.

Habiendo hecho un breve análisis de cómo afectan los parámetros a las estadísticas, ahora se analizará como afectaría el manejo de writes a las estadísticas.

Ahora nos enfocaremos en el archivo *test3.json*, cuyos resultados de la simulación son los siguientes:

Programa	HRT	HRC	PF	SO	SI	ST	TT	CT	MT
Solitare	0.0	0.0	1	0	0	0	48	192	1200
Minesweeper	0.0	0.0	1	0	0	0	96	384	1900
Full Tilt! Pinball	0.6	0.2	2	0	0	0	192	896	3800
Solitare Spider	0.33	0.0	2	0	0	0	128	576	3100

Los resultados mostrados anteriormente son con el manejo de writes por defecto de la simulación: *Write-Back*.

A continuación se modificará este manejo por uno tipo *Write-Trough* y comparán los resultados.

Programa	HRT	HRC	PF	SO	SI	ST	TT	CT	MT
Solitare	0.0	0.0	1	0	0	0	48	192	1200
Minesweeper	0.0	0.0	1	0	0	0	96	384	1900
Full Tilt! Pinball	0.6	0.2	2	0	0	0	192	896	4300
Solitare Spider	0.33	0.0	2	0	0	0	128	576	3100

Si bien los resultados en esta prueba no son impactantes debido al bajo hit rate de la caché, todavía se pueden ver los efectos esperados: aumemento en el tiempo total de memoria, como se puede ver en el caso del tercer programa, el cual es el único con un caché hit rate mayor a cero. Esto se puede explicar ya que, con *Write-Through*, calquier escritura en la caché significa una escritura en Memoria física, mientras que con *Write-Back* sólo se escribe en memoria al momento de reemplazar una dirección en la caché. Se espera que con altos hit-rate de la caché el aumento de tiempo total de memoria sea mucho mayor.

2 Bibliografía

Para el presente informe se utilizaron, principalmente, los apuntes del curso IIC2343 'Arquitectura de Computadores' creados por Alejandro Echeverría y Hans-Albert Löbel. Capítulos 'Memoria Caché' y 'Multiprogramación', accesibles mediante:

https://github.com/IIC2343/syllabus/tree/master/Apuntes

También se obtuvo información del foro de ayuda de dicho curso:

https://github.com/IIC2343/syllabus/issues