Lab 3: Evaluación de Memoria Caché

Resultados Loop-merge:

	Dr	**D1mr**	**DLmr**	**Dw**	**D1mw**	**DLmw**
16 KiB	1,430,105 rd	26,442 rd	26,200 rd	211,329 wr	13,070 wr	13,045 wr
32 KiB	1,430,105 rd	26,321 rd	26,207 rd	211,329 wr	13,062 wr	13,044 wr
64 KiB	1,430,105 rd	26,247 rd	26,198 rd	211,329 wr	13,052 wr	13,042 wr

Resultados Loop-merge-opt:

	Dr	**D1mr**	**DLmr**	**Dw**	**D1mw**	**DLmw**
16 KiB	1,230,104 rd	13,942 rd	13,700 rd	211,328 wr	13,070 wr	13,045 wr
32 KiB	1,230,104 rd	13,821 rd	13,707 rd	211,328 wr	13,062 wr	13,044 wr
64 KiB	1,230,104 rd	13,747 rd	13,698 rd	211,328 wr	13,052 wr	13,042 wr

Como podemos ver en las tablas superiores, las diferencias del tamaño de la caché en ambos casos no influencian los resultados de las operaciones de lectura y escritura de forma considerable. Por otra parte, podemos apreciar como en el caso de unir ambos bucles en uno los resultados son más eficientes en las operaciones de lectura que en el caso de tener dos bucles por separado. Esto ocurre en todos los niveles de la caché. Sin embargo, las operaciones de escritura se mantienen iguales en ambos casos. Esto es normal, puesto que no se escribe contenido nuevo a pesar de fusionar los bucles.

En cuanto a los fallos de lectura, podemos apreciar como en el caso del bucle optimizado son aproximadamente la mitad a los del programa sin optimizar, bajando la tasa de fallo de lectura de un 1.8% a un 1.1%.

Resultados soa:

	Dr	**D1mr**	**DLmr**	**Dw**	**D1mw**	**DLmw**
16 KiB	1,230,358 rd	51,737 rd	51,179 rd	5,011,464 wr	100,625 wr	100,568 wr
32 KiB	1,230,358 rd	51,467 rd	51,179 rd	5,011,464 wr	100,599 wr	100,569 wr
64 KiB	1,230,358 rd	51,343 rd	51,179 rd	5,011,464 wr	100,589 wr	100,569 wr

Resultados aos:

	Dr	**D1mr**	**DLmr**	**Dw**	**D1mw**	**DLmw**
8 KiB	1,230,105 rd	51,707 rd	51,166 rd	211,328 wr	25,597 wr	25,541 wr
16 KiB	1,230,105 rd	51,448 rd	51,166 rd	211,328 wr	25,572 wr	25,542 wr
32 KiB	1,230,105 rd	51,328 rd	51,166 rd	211,328 wr	25,562 wr	25,543 wr

Al igual que en los ejemplos anteriores, el tamaño de la caché de ambos casos no influencian los resultados de las lecturas y escrituras. Ambos tienen prácticamente el mismo número de operaciones de lectura en todos los niveles de la caché. Por otra parte, los resultados de las operaciones de escritura son mucho más favorables para el segundo caso que se basa en las estructuras de arrays en vez de en los arrays de estructuras, disminuyendo la cantidad de operaciones de escritura de aproximadamente 5 millones a 211.000. Sin embargo, si apreciamos la tasa de fallo de escritura, podemos apreciar que en el caso del *SOA* es un 2.0%, mientras que en el caso del *AOS* asciende esta tasa a un 12.1%.

Conclusión

No hemos tenido muchos problemas a la hora de realizar este laboratorio, salvo en la realización del script, que tuvimos ciertos fallos al momento de hacerlo pero conseguimos arreglarlos con ayuda de los profesores. Lo único que no conseguimos fue aislar la salida de Valgrind a documentos de texto a parte, se quedó todo el output en el *slurm*. Además el laboratorio muestra perfectamente como distintos cambios al código, vistos en la teoría de clase, pueden generar mejoras en el rendimiento y más especificamente en que operaciones.