Reentrega TP2: Simulador de Cache

Santiago Alvarez Juliá, *Padrón Nro. 99522*
75.42 Taller de Programación
Facultad de Ingeniería, Universidad de Buenos Aires

Octubre 2018

Índice general

0.1. Introducción

En esta reentrega del informe del TP2, explicaré como solucioné los errores que cometí en la primera entrega y algunas sugerencias que fueron propuestas por el corrector.

0.2. Main

Cambié el parseo del archivo de configuración como sugería el corrector, sabiendo que std::getline devuelve un file que tiene implementado el operador bool() que devuelve true si llego al final del archivo y false en caso contrario. Pasé la determinación de cual hijo de Cache es el correcto al constructor de Cache_Protected para que sea RAII (pido memoria al heap en el constructor y la libero en el destructor). También como sugirió el corrector, guardo una referencia al std::map con la info del archivo de configuración en Cache (lo recibe como parámetro al construirse cuando una de sus clases hijas se contruye). Además cambié la implementacion de ThreadCache (le agregué el contructor por movimiento) para no utilizar el heap al lanzar los hilos.

0.3. Cache

Eliminé los set_data y pase lo que implementaban al constructor para que sea RAII. Ahora hice solamente virtual al método process_memory_address ya que antes reimplementaba los métods sin agregarle funcionalidad extra (lo cual estaba mal). En particular para el cache_Associative_Lru cambié la implementacion del método process_memory_address y uno de sus atributos, el std::map que tenía como valores iteradores. Como me remarcó el corrector, anteriormente andaba de casualidad. Ahora el std::map tiene como valores boolean (no tienen uso, podría poner cualquier tipo acá) por lo que averiguar si un tag genera HIT o MISS es O(log n) ya que utilizo el método find de std::map. En el caso de que haya un HIT ahora tengo que recorrer la std::deque para poder borrarlo y volver a encolarlo, lo que tiene como orden de complejidad O(n) (siempre siendo n el tamaño de la std::deque).

0.4. FunctorCache

Para empezar le cambié el nombre a thread_Cache ya que por definición 'FunctorCache' no es un functor ya que no define el método operator () y podría confundir su nombre. Además ahora abro el archivo con las direcciones

de memoria en su constructor (también cambié el flag que indicaba que dicho archivo era binario ya que en el enunciado estaba aclarado que éste era un archivo de texto).