Universidad de Buenos Aires - FIUBA
66.20 Organización de Computadoras
Trabajo práctico 2: Introducción a caches
1^{er} cuatrimestre de 2020

`$Date: 2020/06/01 00:04:30 $`

1. Objetivos

Estudiar el comportamiento de diversos sistemas de memoria cache utilizando una serie de escenarios de análisis o *benchmarks* descriptos a continuación.

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes, un informe impreso de acuerdo con lo que mencionaremos en la sección 6, y con una copia digital de los archivos fuente necesarios para compilar el trabajo.

4. Descripción

En este trabajo estudiaremos el comportamiento de una serie de configuraciones de sistemas de memoria cache, analizando la ejecución de programas de prueba o *benchmarks* en forma similar a lo estudiado en la práctica del día martes 19/5 [1].

Nombre	Tipo	Comentario	Parámetros `cachegrind`
C1	DM	asociatividad trivial	`--I1=64,1,16 --D1=64,1,16 --LL=64,1,16`
C2	2WSA	*2-way set-associative*	`--I1=64,2,16 --D1=64,2,16 --LL=64,2,16`
C3	4WSA	*4-way set-associative*	`--I1=64,4,16 --D1=64,4,16 --LL=64,4,16`

Cuadro 1: configuraciones para el sistema de memoria.

A lo largo de este TP, adoptaremos las 3 configuraciones para el nivel 1 del sistema de memoria cache indicadas en el cuadro 1.

En todos los casos la capacidad total será de 64 bytes, y el tamaño de línea 16 bytes. Para cada una de estas configuraciones, deberá estudiarse el comportamiento de las mismas al ejecutar una serie de 4 programas MIPS32 disponibles en [2]:

- **benchmark-b0**
- **benchmark-b1**
- **benchmark-b2**
- **benchmark-b3**

Para ello, deberá usarse el entorno QEMU del trabajo anterior [3], y la el programa cachegrind [4], una herramienta de *profiling* y simulación de sistemas de memoria cache multinivel que forma parte de la suite de software Valgrind [5].

4.1. Instalación de `cachegrind`

Debido a fallas en la versión de `cachegrind` suministrada dentro de la distribución de Linux que usamos en el TP, en este trabajo será necesario instalar una versión más reciente de esta herramienta en form manual. Para ello basta ejecutar el comando en la consola MIPS32:

```
$ gzip -dc valgrind-mips32-debian-stretch.tar.gz | (cd /opt/; tar -xvf -)
```

4.2. Funcionamiento de `cachegrind`

Para validar que la herramienta está funcionado, podemos compilar y ejecutar el ejemplo suministrado en `/opt/valgrind/share/fiuba/01-holamundo.S`:

```
$ cc -Wall -g -o /tmp/01-holamundo /opt/valgrind/share/fiuba/01-holamundo.S
$ /opt/valgrind/bin/valgrind --tool=cachegrind /tmp/01-holamundo
...
Hola mundo.
...
```

(Notar que en el ejemplo de arriba sólo hemos mostrado la salida propia del programa, y hemos suprimido las líneas generadas por el propio `cachegrind`). Este último comando ejecuta el binario `01-holamundo` dentro de la herramienta de profiling del sistema de memoria, y toma nota de la actividad realizada por el cache en el archivo `cachegrind.out.$pid`, donde `$pid` representa el número de proceso UNIX que tenía el proceso en el momento de realizar la simulación (en este caso `$pid` vale `3470`).

Para acceder a la información de *profiling* del sistema de memoria, basta con ejecutar el programa `cg_annotate`, indicando la ubicación del archivo con el código fuente del programa, y de esa manera poder acceder a las anotaciones línea por línea de la actividad del sistema de cache en nuestros programas MIPS32:

```
$ /opt/valgrind/bin/cg_annotate cachegrind.out.3470 \
          /opt/valgrind/share/fiuba/01-holamundo.S
...
--------------------------------------------------------------------------------
```

```
-- User-annotated source: /opt/valgrind/share/fiuba/01-holamundo.S
--------------------------------------------------------------------------------
Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw

-- line 4 ----------------------------------------
 .    .    .  .    .    .  .    .    .
 .    .    .  .    .    .  .    .    .   .text
 .    .    .  .    .    .  .    .    .   .align 2
 .    .    .  .    .    .  .    .    .
 .    .    .  .    .    .  .    .    .   .globl main
 .    .    .  .    .    .  .    .    .   .ent main
 .    .    .  .    .    .  .    .    .  main:
 .    .    .  .    .    .  .    .    .   .set noreorder
 3    1    1  0    0    0  0    0    0   .cpload t9
 .    .    .  .    .    .  .    .    .   .set nomacro
 1    0    0  0    0    0  0    0    0   addiu sp, sp, -24
 1    1    1  0    0    0  1    0    0   sw fp, 20(sp)
 1    0    0  0    0    0  0    0    0   move fp, sp
 1    0    0  0    0    0  1    0    0   .cprestore 0
 .    .    .  .    .    .  .    .    .
 1    0    0  0    0    0  0    0    0   li a0, 1
 2    0    0  1    0    0  0    0    0   la a1, msg
 1    0    0  0    0    0  0    0    0   li a2, 12
 1    0    0  0    0    0  0    0    0   li v0, SYS_write
 1    1    1  0    0    0  0    0    0   syscall
 1    0    0  0    0    0  0    0    0   nop
 .    .    .  .    .    .  .    .    .
 1    0    0  0    0    0  0    0    0   move sp, fp
 1    0    0  1    0    0  0    0    0   lw fp, 20(sp)
 1    0    0  0    0    0  0    0    0   addiu sp, sp, 24
 .    .    .  .    .    .  .    .    .
 1    0    0  0    0    0  0    0    0   move v0, zero
 1    0    0  0    0    0  0    0    0   jr ra
 .    .    .  .    .    .  .    .    .   .end main
 .    .    .  .    .    .  .    .    .
 .    .    .  .    .    .  .    .    .   .rdata
 .    .    .  .    .    .  .    .    .  msg:
 .    .    .  .    .    .  .    .    .   .asciiz "Hola mundo.\n"

--------------------------------------------------------------------------------
Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
--------------------------------------------------------------------------------
19    3    3  2    0    0  2    0    0  events annotated
```

5. Desarrollo

Para cada combinación de los *benchmarks* y configuraciones del sistema de memoria presentados en la sección 4 (12 combinaciones o casos de análisis en total), deberá realizarse un análisis detallado del *profiling* realizado con Valgrind y `cg_annotate`, analizando la salida línea por línea, como vimos en la sección 4.2.

En el informe del trabajo práctico, para caso de análisis deberá incluirse:

a) Detalle de todos los comandos usados para recolectar los datos de cada una de las corridas:

invocación de `valgrind`, `cg_annotate`, etc.

b) Incorporar las anotaciones línea por línea de `cg_annotate` correspondiente al archivo `.S` del *benchmark* analizado.

c) Para el cada línea del *benchmark*, explicar en detalle porqué se produce la cantidad de desaciertos indicada para L1D y L1I.

d) Calcular la cantidad total de accesos a memoria, aciertos y desaciertos realizada por los caches L1.

e) Calcular los *miss-rates* para L1D y L1I asociado a esa combinación de *benchmark* y configuración del sistema de memoria.

En todos los casos, La ejecución deberá realizarse en el entorno MIPS32 simulado por QEMU. Asimismo, como en el TP anterior deberá usarse el modo 1 del sistema operativo para manejo de acceso no alineado a memoria [7].

6. Informe

El informe deberá incluir:

- Análisis detallado de cada uno de los *benchmarks*, siguiendo los lineamientos descriptos en las sección 5, respetando la estructura de sub-secciones y títulos.
- El código fuente de todos los programas analizados en el TP.
- Instrucciones de compilacióm y ejecución de cada caso.
- Este enunciado.

7. Entrega de TPs

La entrega de este trabajo deberá realizarse usando el campus virtual de la materia [8]. Asimismo, en todos los casos, estas presentaciones deberán ser realizadas durante los días martes. El *feedback* estará disponible de un martes hacia el otro, como ocurre durante la modalidad presencial de cursada.

Por otro lado, la última fecha de entrega y presentación para esta trabajo será el martes 16/6.

Referencias

[1] Organización de Computadoras - Caches y benchmarks. Facultad de Ingeniría, Universidad de Buenos Aires. Ejercicio explicado en la clase del martes 19/5, primer cuatrimestre 2020. `https://drive.google.com/file/d/1sA4Rzp-HtZ4wgA5sXonY9mMDEqILJ1mM/view`.

[2] Código fuente de los *benchmarks* para analizar en este TP. `https://drive.google.com/drive/folders/1ooE_fWKaflZZTbkN649jbJkKrHaHfDVW`.

[3] Enunciado del Trabajo Práctico 1, primer cuatrimestre de 2020. `https://drive.google.com/drive/folders/1RZNflRb6sG8nqsUAVxS2Ch1pNnEJCxxi`.

[4] Cachegrind: a cache profiler. http://valgrind.org/docs/manual/cg-manual.html.

[5] Valgrind: programming tool for memory debugging, memory leak detection, and profiling. https://valgrind.org/.

[6] Binarios de Valgrind para correr en QEMU MIPS32. https://drive.google.com/drive/folders/1ooE_fWKaflZZTbkN649jbJkKrHaHfDVW.

[7] Controlling the kernel unalignment handling via debugfs (Linux/MIPS wiki). https://www.linux-mips.org/wiki/Alignment.

[8] Aula Virtual - Organización de Computadoras 86.37/66.20 - Curso 1 - Turno Martes. https://campus.fi.uba.ar/course/view.php?id=649