Universidad de Buenos Aires - FIUBA
66.20 Organización de Computadoras
Trabajo práctico 2: Introducción a caches
2^o cuatrimestre de 2020

`$Date: 2020/11/15 12:57:47 $`

1. Objetivos

Estudiar el comportamiento de los 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 un programa que permite multiplicar matrices cuadradas con cachegrind `cachegrind`.

Nombre	Tipo	Asociatividad	Parámetros `cachegrind`
C1	DM	Trivial	`--I1=32768,4,32 --D1=32768,1,32`
C2	2WSA	*2-WSA*	`--I1=32768,4,32 --D1=32768,2,32`
C3	4WSA	*4-WSA*	`--I1=32768,4,32 --D1=32768,4,32`

Cuadro 1: configuraciones para el sistema de memoria.

A lo largo de este TP, adoptaremos las 3 configuraciones para el nivel 1 y 2 del sistema de memoria cache indicadas en el cuadro 1.

En todos los casos la capacidad total será de 32 kbytes, y el tamaño de línea 32 bytes. Para cada una de estas configuraciones, deberá estudiarse el comportamiento de las mismas al ejecutar una implementación del algoritmo naïve de multiplicación de matrices cuadradas.

Para ello, deberá usarse el entorno QEMU del trabajo anterior [2], y la el programa cachegrind [3], una herramienta de *profiling* y simulación de sistemas de memoria cache multinivel que forma parte de la suite de software Valgrind [4].

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

Usando las herramientas, deberá ejecutarse el algoritmo naïve de multiplicación de matrices repetidas veces, con tamaño de matriz progresivamente creciente: `n = 0, 2, 4, 8, ...`, en donde `n` representa la cantidad de filas y columnas de la matriz.

Deberán estudiarse los accesos que el algoritmo realiza sobre el sistema de memoria para acceder a los elementos de la matriz y calcular los productos internos de los vectores fila y columna.

Los resultados deberán representarse en un diagrama como el de la figura 1 que permita representar en la cantidad de desaciertos del cache L1D en función de `n`, como así también la tasa misses asociada.

Asimismo, cada grupo deberá realizar una predicción teórica del comportamiento del cache, el cual deberá contrastarse con las mediciones de cachegrind y documentadas en el informe (coincidan o no).

En este trabajo práctico no se requiere estudiar el comportamiento de otros niveles de cache (sólo L1D).

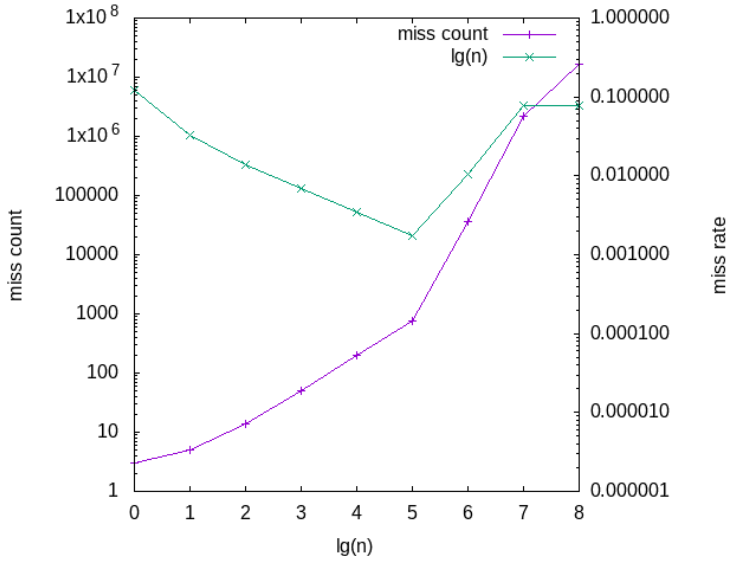

Figura 1: representación de cantidad y tasa de desaciertos para un dada combinación de algoritmo y configuración de cache.

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) Explicar en detalle porqué se produce la cantidad de desaciertos indicada sobre L1D para cada una de las configuraciones del sistema de memoria del cuadro 1[1].

d) Calcular la cantidad total de accesos a memoria, aciertos y desaciertos realizada por el cache L1D.

e) Calcular las tasa de desacierto para L1D asociada a cada combinación de *benchmark* y configuración del sistema de memoria.

f) Contrastar detalladamente los resultados de los cálculos con las simulaciones. **Justificar todo**.

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 [6].

6. Informe

El informe deberá incluir:

[1] Optativamente, se sugiere explorar también otras asociatividades, tamaños de bloque y variantes del algoritmo y contexo de ejecución

- Análisis detallado de cada uno de los *benchmarks*, siguiendo los lineamientos descriptos en las sección 5.
- 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 únicamente a través del campus virtual de la materia [7] dentro del plazo de tiempo establecido.

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 es el martes 1/12.

Referencias

[1] Implementación de referencia para el algoritmo naïve de multiplicación de matrices. `https://drive.google.com/file/d/1gEVZ8d7IVMu9R3QUBlwmjQTXjXYEeJVJ/view?usp=sharing`

[2] Enunciado del Trabajo Práctico 1, segundo cuatrimestre de 2020. `https://drive.google.com/file/d/1dtWzlAwxxpLqwifaCwJ3_0WyYwvoavUd/view?usp=sharing`.

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

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

[5] Binarios de Valgrind para correr en QEMU MIPS32. `https://drive.google.com/file/d/1n4_b5xHjaA8dAmEiileoAOlZjjsunRWa/view?usp=sharing`.

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

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