Organización y Arquitectura de Computadoras
2016-2
Práctica 11: Desempeño de cachés

Profesor: José de Jesús Galaviz Casas
Ayudante: Roberto Monroy Argumedo

Publicada: Mayo 23, 2016.
Límite de entrega: Junio 12, 2016.

1. Objetivos

Generales:

- El alumno aprenderá a realizar un analisis de desempeño de cachés.

Particulares:

Al finalizar la práctica el alumno conocerá:

- Los principios básicos del funcionamiento de cachés.
- Un mecanismo para medir el desempeño de cachés.

2. Requisitos

- **Conocimientos previos:**
  - Jerarquía de memoria:
    - Principio de localidad.
    - Colocación y remplazo de bloques en caché.
    - Localización de bloques en caché.
    - Desempeño de cachés.
  - Programación en el lenguaje C:
    - Tipos de datos.
    - Estructuras de control.
    - Estructuras y uniones
    - Arreglos y apuntadores.

    - Entrada y salida.

- **Tiempo de realización sugerido:**
  10 horas.

- **Número de colaboradores:**
  Equipos de 2 integrantes.

- **Software a utilizar:**

  - *GCC.*
  - *Pin - A Binary Instrumentation Tool.*

3. Planteamiento

En esta práctica se desarrollará un simulador básico de cachés, el programa recibirá la configuración de uno o más niveles de cachés y una traza de accesos a memoria, el programa deberá simular los accesos a memoria con el objetivo de analizar el desempeño de la configuracion dada.

4. Desarrollo

4.1. Descripción del simulador

El programa de deberá simular los siguientes aspectos:

- Se recibirá una traza de memoria en donde cada entrada constará del tipo de acceso: lectura o escritura y la dirección de memoria accedida.

- Se deberá determinar el tiempo que toma cada uno de los accesos de memoria simulando los fallos y aciertos.

- No es necesario tener espacio para los datos en el simulador, por cada bloque bastará tener los campos necesarios para el manejo del caché: etiqueta, estampa de tiempo, etc.

- Se podrá contar con uno o más niveles de caché.

- Los tiempos de acierto de cada caché y la penalización por fallos de acceso a la memoria en el último nivel serán dados por el usuario del simulador.

- Las penalizaciones por fallo de los niveles intermedios de cachés serán calculados dependiendo del tiempo tomado por el siguiente caché en traer el dato.

- Los cachés serán asociativos de conjuntos de n posibilidades, ya que otros modos (mapeo directo y completamente asociativo) son subcasos de éste.

- Se implementarán dos modos de remplazo de bloques: menos reciente usado) y remplazo aleatorio.
- Se implementarán dos modos de escritura: *write-through* y *write-back*.
- Todas las escrituras serán *write allocate*.

Al finalizar la ejecución, el programa deberá entregar los siguientes datos:

- Tasa de aciertos por nivel.
- Tasa de fallos por nivel.
- Tiempo total de ejecución.
- Tiempo promedio de acceso a memoria.

4.2. Traza de accesos a memoria

Pin es un *framework* desarrollado por *Intel* que permite insertar dinámicamente código arbitrario en ejecutables, es decir, en tiempo de ejecución. Fue desarrollado para la creación de herramientas de análisis de arquitecturas de computadoras pero actualmente se usa también en otras áreas como seguridad o cómputo paralelo. El *framework* puede ser utilizado tanto en Windows como en Linux (además de otras plataformas).

Para obtener las trazas de accesos a memoria empleadas por el simulador, usaremos un código de ejemplo incluido en el *framework* llamado `pinatrace`.

4.2.1. Instrucciones para compilar `pinatrace` en Linux:

1. Descarga el software de la página oficial del *framework* [1].
2. Descomprime el paquete descargado.
3. Cámbiate al directorio `source/tools/ManualExamples` dentro del directorio recién descomprimido.
4. Ejecuta el siguiente comando para compilar el archivo:
   - Si tu arquitectura es de 32 bits:
     `$ make obj-ia32/pinatrace.so TARGET=ia32`
   - Si tu arquitecura es de 64 bits:
     `$ make obj-intel64/pinatrace.so TARGET=intel64`

4.2.2. Instrucciones para obtener una traza de memoria:

Para obtener la traza de memoria se puede utlizar cualquier archivo ejecutable, en éste ejemplo usaremos el del comando `ls`.

1. Desde el directorio `source/tools/ManualExamples` ejecuta el siguiente comando:

   - Si tu arquitectura es de 32 bits:
     `$ ../../../pin -t obj-ia32/pinatrace.so -- /bin/ls`
   - Si tu arquitectura es de 64 bits:
     `$ ../../../pin -t obj-intel64/pinatrace.so -- /bin/ls`

2. El programa escribirá un archivo con nombre `pinatrace.out` con la traza de la memoria, la primer columna es la dirección de la instrucción que solicito el acceso a memoria, la segunda columna es el tipo de acceso y la tercer columna es la dirección de memoria solicitada:

```
0x7f8fb63a2d93: W 0x7ffc684a8078
0x7f8fb63a6ad0: W 0x7ffc684a8070
0x7f8fb63a6ad4: W 0x7ffc684a8068
0x7f8fb63a6ad6: W 0x7ffc684a8060
0x7f8fb63a6ad8: W 0x7ffc684a8058
0x7f8fb63a6ada: W 0x7ffc684a8050
0x7f8fb63a6adf: W 0x7ffc684a8048
0x7f8fb63a6af6: W 0x7f8fb65c5c40
0x7f8fb63a6afd: R 0x7f8fb65c5e50
0x7f8fb63a6b07: R 0x7f8fb65c5fb0
```

3. Como la primer columna no será de utilidad para el simulador, puedes eliminarla modificando el código fuente de `pinatrace` o con el comando `cut`:

   ```
   $ cut -f 2-3 -d ' ' pinatrace.out > traza
   ```

5. Entrada

El simulador recibirá por la línea de comandos los siguientes parámetros:

1. Latencia de RAM. Tiempo necesario para traer un dato de la memoria RAM al último nivel de cahé.

2. Archivo con la traza de acceso a memoria.

3. Uno o más archivos con la configuración de los cachés, un archivo por cada caché. El orden definirá los niveles, el primero en aparecer en la línea será el más cercano al procesador y el último en aparecer será el más cercano a la memoria RAM.

5.0.1. Formato de archivo de configuración de caché:

El archivo contentrá los siguientes parámetros, uno por línea:

- Tamaño del caché en kilobytes. Un entero positivo potencia de 2. El tamaño no incluye los bits necesarios para el manejo del caché como: etiqueta, estampa de tiempo, etc.
- Tamaño del bloque. Un entero positivo potencia de 2.
- Tiempo de acierto (*Hit-time*). Un real positivo.
- Asociatividad. Un entero positivo.
- Tipo de remplazo de bloques. '0' para aleatorio y '1' para el usado menos recientemente.
- Tipo de escritura: '0' para *write-througt* y '1' para *write-back*.

6. Salida

El simulador entregará un archivo con los siguientes datos:

- Tasa de aciertos por nivel.
- Tasa de fallos por nivel.
- Tiempo total de ejecución.
- Tiempo promedio de acceso a memoria.

7. Variables libres

El alumno propondrá un esquema de cachés para medir su desempeño.

8. Procedimento

La entrega constará de los siguientes elementos:

1. El archivo de código fuente escrito en el lenguaje de programación C con la solución al ejercicio 1. El programa debe estar completamente documentado y cumplir con las convenciones de código.
2. La traza usada para evaluar el desempeño de los cachés.
3. El reporte con el análisis de desempeño.

9. Ejercicios

1. Desarrolla el simulador descrito en la sección de 4.1.
2. Obtén una traza de memoria.
3. Con la misma traza, corre la simulación con cada uno de los siguientes esquemas:

   a) **Esquema A:**
   Latencia de RAM: 64.4 ns.

   Caché L1

Tamaño:	32 KB
Tamaño del bloque:	64 Bytes
Hit-time:	1.2 ns
Asociatividad:	8 posibildades
Remplazo de bloques:	Usado menos recientemente
Escritura:	Write-back

   Caché L2

Tamaño:	265 KB
Tamaño del bloque:	64 Bytes
Hit-time:	3.3 ns
Asociatividad:	8 posibilidades
Remplazo de bloques:	Usado menos recientemente
Escritura:	Write-through

   b) **Esquema B:**
   Latencia de RAM: 64.4 ns.

   Caché L1

Tamaño:	32 KB
Tamaño del bloque:	64 Bytes
Hit-time:	1.2 ns
Asociatividad:	8 posibildades
Remplazo de bloques:	Usado menos recientemente
Escritura:	Write-back

   Caché L2

Tamaño:	265 KB
Tamaño del bloque:	64 Bytes
Hit-time:	3.3 ns
Asociatividad:	8 posibilidades
Remplazo de bloques:	Usado menos recientemente
Escritura:	Write-back

   c) **Esquema C:**
   Latencia de RAM: 61.5 ns.

Caché L1

Tamaño:	64 KB
Tamaño del bloque:	64 Bytes
Hit-time:	1 ns
Asociatividad:	4 posibildades
Remplazo de bloques:	Aleatorio
Escritura:	Write-back

d) **Esquema D:**
Latencia de RAM: 61.5 ns.

Caché L1

Tamaño:	64 KB
Tamaño del bloque:	64 Bytes
Hit-time:	1 ns
Asociatividad:	8 posibildades
Remplazo de bloques:	Aleatorio
Escritura:	Write-back

4. De acuerdo a las observaciones, propón un nuevo esquema en el que consideres mejorará el desempeño. Cuentas con los siguientes opciones de memoria:
   - L1 con hit-time de 1 ns.
   - L2 con hit-time de 3 ns.
   - L3 con hit-time de 10 ns.
   - Latencia de RAM: 61.5 ns.
5. Elabora un reporte en donde analices los resultados obtenidos. Incluye gráficas de las tasas de aciertos, tasas defallos, tiempos de ejecución y tiempo promedio de acceso a memoria.

10. Preguntas

1. ¿Cuáles fueron los factores determinantes en el desempeño del esquema de cachés con mejor rendimiento?
2. ¿Qué consideraciones tomaste en cuenta para diseñar el esquema de cachés que propusiste? ¿Resultó tener mejor desempeño?

11. Bibliografía

[1] Intel. *Pin - A Dynamic Binary Instrumentation Tool*. Consultado: 16 de abril, 2016. Publicado: 2012. URL: `https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool/`.

[2] Brian W. Kernighan. *The C Programming Language*. 2nd. Prentice Hall Professional Technical Reference, 1988.

[3] David A. Patterson y John L. Hennessy. *Computer Organization and Design, Fifth Edition: The Hardware/Software Interface*. 5th. Morgan Kaufmann Publishers Inc., 2013.