

Práctica 5: Caché

Gustavo Romero López

Arquitectura y Tecnología de Computadores

8 de enero de 2015

Índice

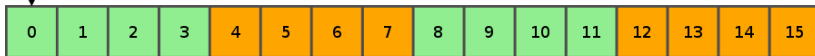
- 1 Índice
- 2 Objetivos
- 3 Tamaño de línea
- 4 Tamaño de caché
- 5 Evaluación
- 6 Enlaces

Objetivos

- Comprender la importancia de la memoria caché mediante el estudio de la misma.
- Nos centraremos en dos de sus parámetros más importantes:
 - Tamaño de línea o **bloque**.
 - **Tamaño de caché**.
- Intentaremos calcularlos para el procesador que utilizamos.
- En versiones modernas de Linux podemos consultar todos los parámetros de la caché mediante la orden `lscpu` o examinando el directorio `/sys/devices/system/cpu/cpu0/cache`.

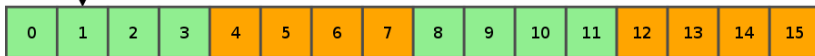
Tamaño de línea

primer acceso: supongamos fallo... trae 0, 1, 2 y 3



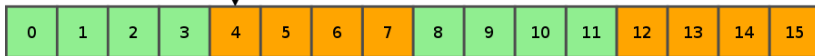
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----

segundo acceso: acierto, 2 está dentro de la misma línea que 0



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----

tercer acceso: fallo, 4 pertenece a otra línea... trae 4, 5, 6 y 7



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----

Tamaño de línea

- Una línea o bloque de caché es la cantidad de información que viaja entre los niveles de caché y la memoria principal.
- Es tan importante que a veces prevalece el acceso a los datos frente a su tratamiento.
- Para comprobar la afirmación anterior mida cuánto tardan en ejecutarse los siguientes bucles dos bucles.
 - El primero hace un pequeño cambio en cada entero.
 - El segundo a uno de cada 16.
- Antes de comprobarlo, ¿cuánto tiempo calcula que debería tardar cada uno?

Tamaño de línea

```
1 #include <array>
2 #include <chrono>
3 #include <iostream>
4 using namespace std::chrono;
5
6 std::array<int, 64*1024*1024> bytes; // 64Mx4B = 256MB
7
8 int main()
9 {
10     auto start = high_resolution_clock::now();
11     for (unsigned i = 0; i < bytes.size(); ++i)
12         bytes[i] |= 0x1;
13     auto stop = high_resolution_clock::now();
14     std::cout << duration_cast<milliseconds>(stop - start).
15         count() << std::endl;
16
17     start = high_resolution_clock::now();
18     for (unsigned i = 0; i < bytes.size(); i += 16)
19         bytes[i] |= 0x1;
20     stop = high_resolution_clock::now();
21     std::cout << duration_cast<milliseconds>(stop - start).
22         count() << std::endl;
23 }
```

Tamaño de línea

- Como deseamos medir el tamaño de línea vamos a generalizar el anterior proceso para todos los tamaños de línea posibles.
- Tenemos que meter el bucle del listado anterior dentro de otro que recorra todos los tamaños de línea posibles.
- Cuánto más ligero sea este bucle mejor se evidenciará la diferencia de tiempos entre cálculo y acceso a memoria.
- Medir tiempos y comparar.
- ¿Los resultados obtenidos se parecen a los esperados?
- El resultado en mi ordenador puede verse en la figura 1.
- En vez de partir de 0, complete el esqueleto: line.cc.
- Makefile genera un gráfico de forma automática.
- Razone que tamaño de línea utiliza su procesador.
- Puede ayudarse con la orden `lscpu`.

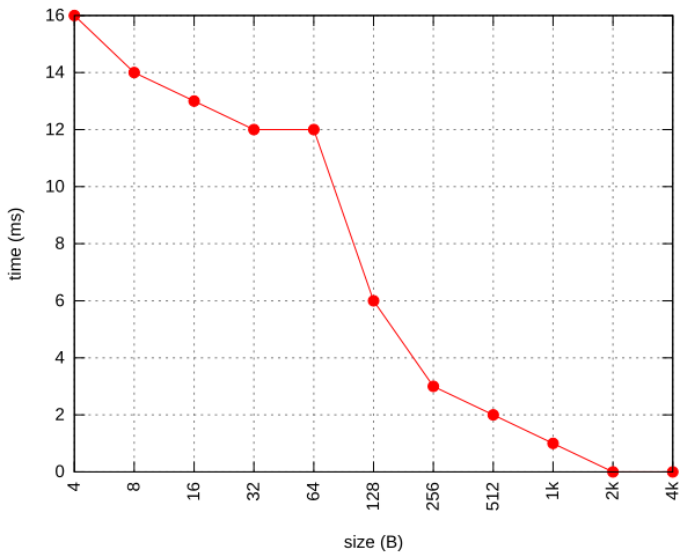


Figura 1: Tamaño de línea.

Tamaño de caché

- Mejor cuanto más grande si no fuese por precio, calor, superficie, consumo,...
- Ya sabemos el tamaño de la línea porque se ha medido en el apartado anterior.
- Para medir el tamaño de caché debemos:
 - Para cada tamaño de caché
 - Crear un vector de dicho tamaño
 - Repetir 1000000 veces.
Realizar una pequeña alteración en una línea.
- **Cuánto más ligero sea este bucle mejor se evidenciará la diferencia de tiempos entre cálculo y acceso a memoria.**
- Medir tiempos y comparar.
- El resultado en mi ordenador puede verse en la figura 2.
- En vez de partir de 0, complete el esqueleto: `size.cc`.
- Makefile genera un gráfico de forma automática.
- ¿Cuántos niveles de caché tiene su procesador? ¿De qué tamaño?

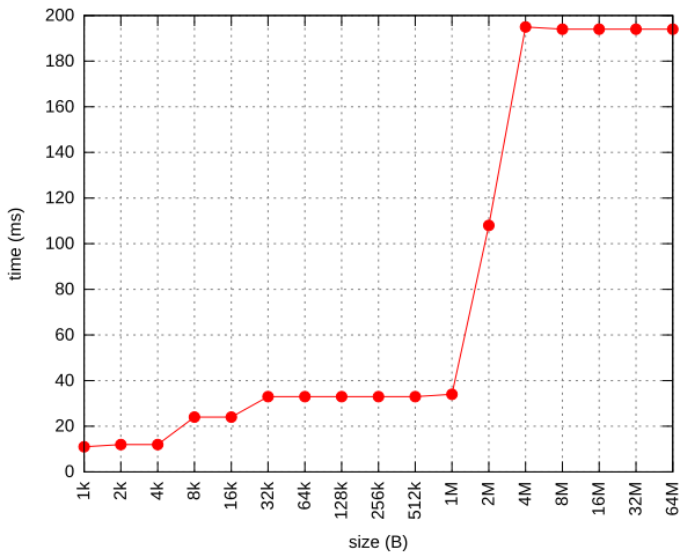


Figura 2: Tamaño de cache.

Evaluación

Para cada uno de los dos parámetros estudiados debe entregar:

- El programa: `line.cc` y `size.cc`.
- El gráfico generado por el Makefile para su CPU: `line.png` y `size.png`.
- Una explicación razonada de los resultados obtenidos.
- Un pantallazo con la ejecución de una de estas tres cosas:
 - `lscpu`
 - `CPUG`
 - `make info`

lscpu

```

Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    2
Core(s) per socket:    2
Socket(s):             1
NUMA node(s):         1
Vendor ID:             AuthenticAMD
CPU family:            21
Model:                 48
Model name:            AMD A10-7700K APU with Radeon(TM) R7 Graphics
Stepping:              1
CPU MHz:               2000.000
CPU max MHz:           3400,0000
CPU min MHz:           2000,0000
BogoMIPS:              6787.77
Virtualización:        AMD-V
L1d cache:             16K
L1i cache:             96K
L2 cache:              2048K
NUMA node0 CPU(s):     0-3

```

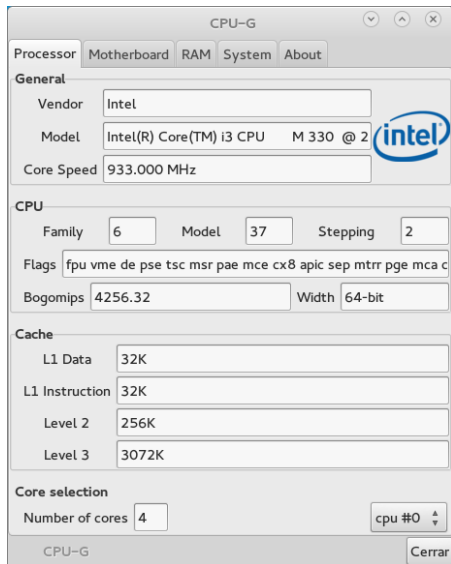


Figura 3: La CPU de mi portatil vista con CPUG

Enlaces de interés

- https://en.wikipedia.org/wiki/CPU_cache
- <http://igoro.com/archive/gallery-of-processor-cache-effects/>
- <http://cpug.sourceforge.net/>