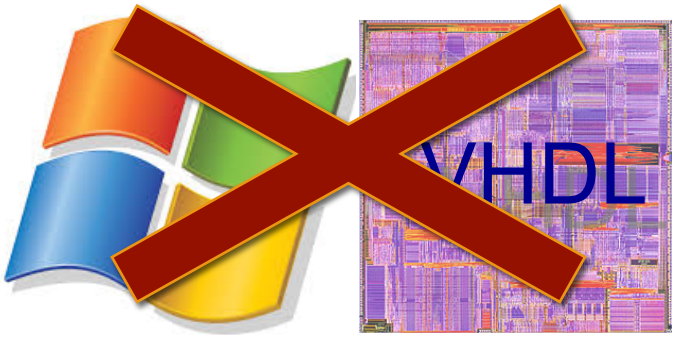
Práctica 3.
Memoria Caché y rendimiento.
Arquitectura de Ordenadores

VÍCTOR MORENO, ALBERTO SÁNCHEZ

ESCUELA POLITÉCNICA SUPERIOR. UAM.

Práctica 3: entorno de de trabajo

- A lo largo de las prácticas 3 y 4 vamos a trabajar utilizando *software* para poner en práctica los conceptos *hardware* de teoría



Práctica 3: material de partida

- **arqo3.c:** código fuente de la librería de manejo de matrices de números en coma flotante
- **arqo3.h**: fichero de cabeceras para la librería de manejo de matrices
- **slow.c**: programa de ejemplo que calcula la suma de los elementos de una matriz
- **fast.c**: programa de ejemplo que calcula la suma de los elementos de una matriz (más eficiente que el anterior).
- **Makefile**: fichero utilizado para la compilación de los ejemplos aportados.

Tiempo de ejecución de un programa

```
int main( int argc, char *argv[])
{
    int n;
    num **m=NULL;
    struct timeval fin,ini;
    num res;

    printf("Word size: %ld bits\n",8*sizeof(num));
    if( argc!=2 )
    {
        printf("Error: ./%s <matrix size>\n", argv[0]);
        return -1;
    }
    n=atoi(argv[1]);
    m=generateMatrix(n);
    if( !m ){
        return -1;
    }

    gettimeofday(&ini,NULL);

    /* Main computation */
    res = compute(m,n);
    /* End of computation */

    gettimeofday(&fin,NULL);
    printf("Execution time: %f\n",
        ((fin.tv_sec*1000000+fin.tv_usec)-(ini.tv_sec*1000000+ini.tv_usec))*1.0/1000000.0);
    printf("Total: %lf\n",res);



    free(m);
    return 0;
}
```

Ejemplo de script que toma datos (I)

```
#!/bin/bash

#genero un fichero con 10000 numeros aleatorios
maxsz=100000
file="random.dat"
rm -f $file
for i in $(seq 1 $maxsz)
do
    echo $RANDOM >> $file
done
```

Ejemplo de script que toma datos (II)

```
#!/bin/bash

sortfile="random.dat"
timefile="time.dat"
rm $timefile

minsz=10000
maxsz=100000
incr=10000
for i in $(seq $minsz $incr $maxsz)
do
    echo "Ordenando $i numeros..."
    head -n $i $sortfile > auxfile
    /usr/bin/time -o aux sort auxfile > /dev/null
    tiempo=$(cat aux | head -n 1 | awk -v FS="u" '{print $1}')
    echo $i $tiempo >> $timefile
done
rm auxfile
rm aux
```

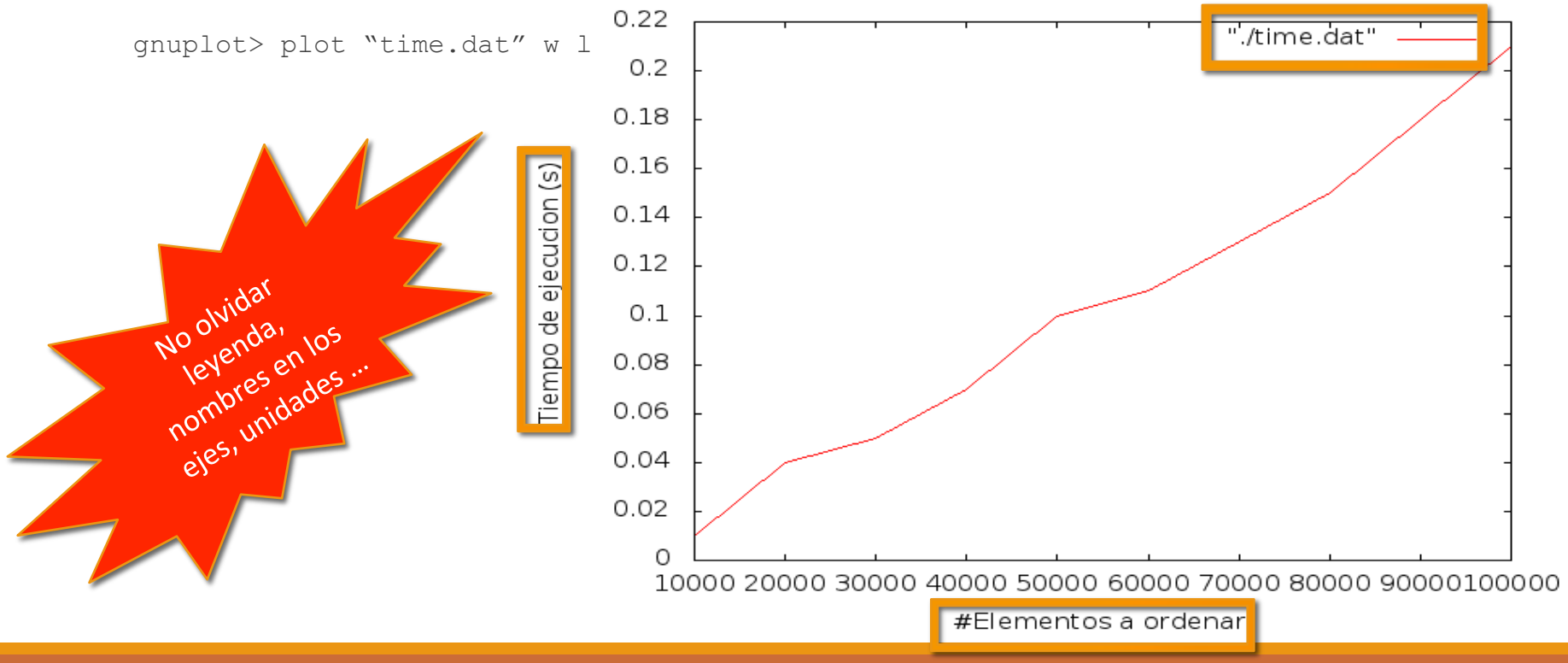
Ejemplo de script que toma datos (III)

```
> gnuplot

gnuplot> set ylabel "Tiempo de ejecucion (s)"

gnuplot> set xlabel "#Elementos a ordenar"

gnuplot> plot "time.dat" w l
```

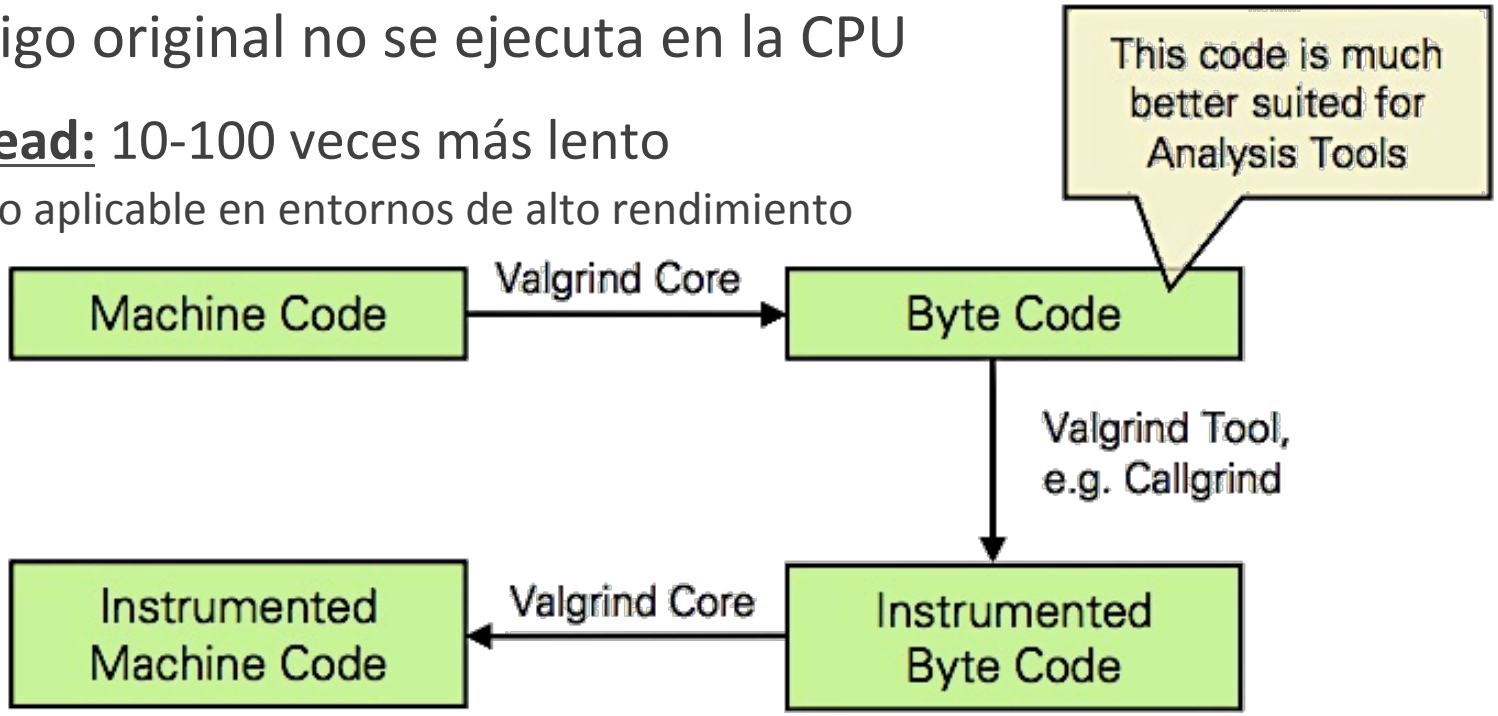


El *framework* de Valgrind (I)

- Conjunto de herramientas basadas en simulación para *debug* y *profiling*
- Valgrind simula una CPU "software"
- Las herramientas implementan sus distintas tareas añadiendo código de análisis
- *Open Source*
- Es un paquete estándar de Linux
- Utilizado por Firefox, OpenOffice, KDE, …
- http://www.valgrind.org

El *framework* de Valgrind (II)

- Recompilación dinámica del código máquina del programa destino en tiempo de ejecución
- El código original no se ejecuta en la CPU
- **Overhead:** 10-100 veces más lento
  - ➜no aplicable en entornos de alto rendimiento



El *framework* de Valgrind (II)

- **Memcheck:** la más común, detección de errores en la gestión de la memoria de un programa
- **Cachegrind:** cache *profiling*, localización de las fuentes de fallos de caché
- **Callgrind:** extensión de *cachegrind*, añade generación del grafo de llamadas
- **Massif:** heap (reservas de memoria) profiler
- **Helgrind:** depuración de *threads*

Ejemplo de uso de *cachegrind*

```
> valgrind --tool=cachegrind ./slow 1000
…
> cg_annotate cachegrind.out.<PID>
```

```
--------------------------------------------------------------------------------
I1 cache:         32768 B, 64 B, 8-way associative
D1 cache:         32768 B, 64 B, 8-way associative
LL cache:         8388608 B, 64 B, 16-way associative
Command:          ./slow 1000
Data file:        cachegrind.out.31409
Events recorded:  Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Events shown:     Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Event sort order: Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Thresholds:       0.1 100 100 100 100 100 100 100 100
Include dirs:
User annotated:
Auto-annotation:  off

--------------------------------------------------------------------------------
        Ir  I1mr  ILmr         Dr      D1mr  DLmr        Dw    D1mw    DLmw
--------------------------------------------------------------------------------
85,189,296 1,124 1,110 33,063,102 1,128,562 2,079 9,019,008 125,742 125,678  PROGRAM TOTALS

--------------------------------------------------------------------------------
        Ir I1mr ILmr        Dr      D1mr DLmr        Dw    D1mw    DLmw  file:function
--------------------------------------------------------------------------------
25,975,792    3    3 8,002,480         0    0 4,001,240       0       0  /build/buildd/eglibc-2.15/stdlib/random_r.c:random_r
19,021,040    5    5 8,010,011         2    2 2,002,013 125,126 125,124  /<ruta>/p3/arqo3.c:generateMatrix
18,009,014    2    2 9,005,006 1,126,001   91 1,001,005       0       0  /<ruta>/p3/slow.c:compute
17,000,000    2    2 6,000,000         0    0 1,000,000       0       0  /build/buildd/eglibc-2.15/stdlib/random.c:random
 4,000,000    1    1 1,000,000         0    0 1,000,000       0       0  /build/buildd/eglibc-2.15/stdlib/rand.c:rand
 1,000,247   28   25 1,000,127         9    5        39       1       1  ???:???
```

Ejemplo de uso de *callgrind* (I)

```
> valgrind --tool=callgrind --cache-sim=yes ./slow 1000
…
> callgrind_annotate --auto=yes callgrind.out.<PID>
```

```
--------------------------------------------------------------------------------
-- Auto-annotated source: /home/victor/Dropbox/arqui_grado/arqui_2013_14/Practicas/p3/slow.c
--------------------------------------------------------------------------------
        Ir        Dr        Dw I1mr      D1mr D1mw ILmr DLmr DLmw

-- line 2 ----------------------------------------
         .         .         .    .         .    .    .    .    .  #include <stdlib.h>
         .         .         .    .         .    .    .    .    .  #include <sys/time.h>
         .         .         .    .         .    .    .    .    .
         .         .         .    .         .    .    .    .    .  #include "arqo3.h"
         .         .         .    .         .    .    .    .    .
         .         .         .    .         .    .    .    .    .  num compute(num **matrix,int n);
         .         .         .    .         .    .    .    .    .
         .         .         .    .         .    .    .    .    .  int main( int argc, char *argv[])
         6         0         4    .         .    .    .    .    .  {
         .         .         .    .         .    .    .    .    .       int n;
         1         0         1    .         .    .    .    .    .       num **m=NULL;
         .         .         .    .         .    .    .    .    .       struct timeval fin,ini;
         .         .         .    .         .    .    .    .    .       num res;
         .         .         .    .         .    .    .    .    .
         9         3         3    .         .    .    .    .    .       printf("Word size: %ld bits\n",8*sizeof(num));
     2,135       582       317  101        22   18  100    8    .  => /build/buildd/eglibc-2.15/stdio-common/printf.c:printf (1x)
       788       248       102    0         7    0    0    7    .  => /build/buildd/eglibc-2.15/elf/../sysdeps/x86_64/dl-trampoline.S:_dl_runti
me_resolve (1x)
         2         1         .    .         .    .    .    .    .       if( argc!=2 )
```

Ejemplo de uso de *callgrind* (II)

```
> valgrind --tool=callgrind --cache-sim=yes ./slow 1000
…
> callgrind_annotate –auto=yes callgrind.out.<PID>
```

```
       7         3         2    .         .    .    .    .    .       printf("Total: %lf\n",res);
   4,377     1,103       695   35         2    3   35    1    3  => /build/buildd/eglibc-2.15/stdio-common/printf.c:printf (1x)
       .         .         .    .         .    .    .    .    .
       .         .         .    .         .    .    .    .    .
       .         .         .    .         .    .    .    .    .
       8         4         3    .         .    .    .    .    .       free(m);
     756       242       102    0        49    5    0   14    4  => /build/buildd/eglibc-2.15/elf/../sysdeps/x86_64/dl-trampoline.S:_dl_runti
me_resolve (1x)
     287        77        40   22         8    1   22    1    .  => /build/buildd/eglibc-2.15/malloc/malloc.c:free (1x)
       1         .         .    .         .    .    .    .    .       return 0;
       4         3         0    0         1    0    0    1    . }
       .         .         .    .         .    .    .    .    .
       .         .         .    .         .    .    .    .    .
       .         .         .    .         .    .    .    .    . num compute(num **matrix,int n)
       4         0         3    1         0    0    1    .    . {
       .         .         .    .         .    .    .    .    .       num sum;
       .         .         .    .         .    .    .    .    .       int i,j;
       .         .         .    .         .    .    .    .    .
   4,005     2,002     1,001    1         0    0    1    .    .       for(i=0;i<n;i++)
       .         .         .    .         .    .    .    .    .       {
 4,005,000 2,002,000 1,001,000    .         .    .    .    .    .               for(j=0;j<n;j++)
       .         .         .    .         .    .    .    .    .               {
14,000,000 6,000,000 1,000,000    0 1,126,000    0    0   90    .                       sum += matrix[j][i];
       .         .         .    .         .    .    .    .    .               }
       .         .         .    .         .    .    .    .    .       }
       1         1         .    .         .    .    .    .    .       return sum;
       4         3         1    0         1    0    0    1    . }
```

callgrind + kcachegrind

- Obtención del grafo de llamadas
  - Permite aplicar diversos criterios

```
> kcachegrind callgrind.out.<PID>
```