

Metodología de la Programación Paralela
Curso 2024/25

Herramientas Hardware y Software

Grado en Ingeniería Informática
Facultad de Informática - Universidad de Murcia

Contenidos

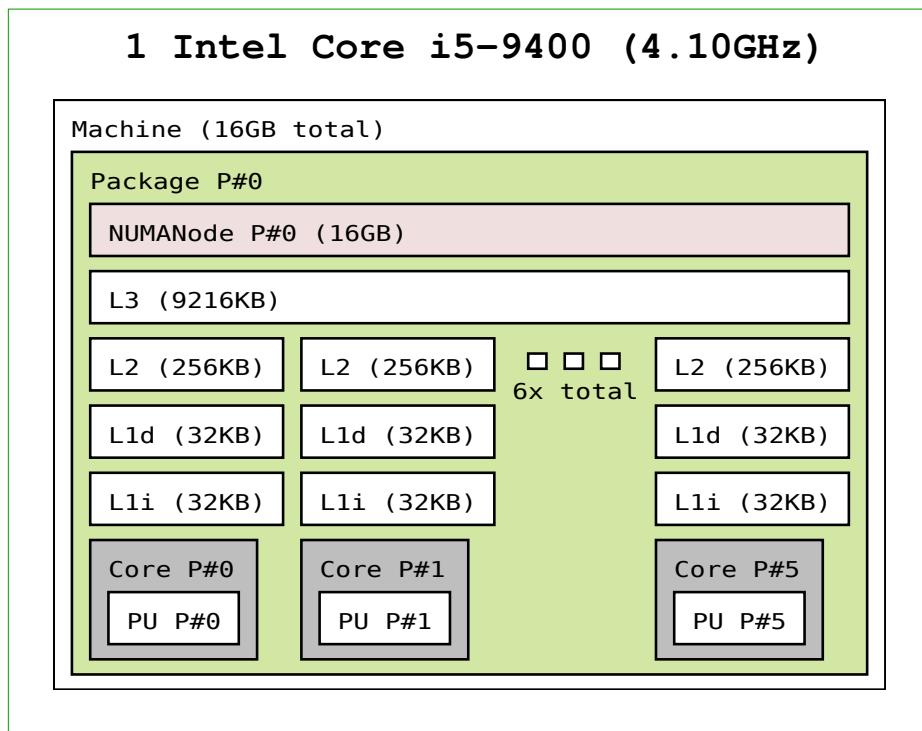
- Herramientas hardware
 - Sistema computacional
- Herramientas software
 - Valgrind

Contenidos

- Herramientas hardware
 - Sistema computacional
- Herramientas software
 - Valgrind

Sistema computacional

- Equipos laboratorio 2.1:



+ 1 GPU Geforce GT 1030 (Pascal)

384 cores - 2 GB

Compilar y ejecutar los
programas de ejemplo:

*Prácticas/Práctica_0/
Sesión_1/Ejemplos*

Contenidos

- Herramientas hardware
 - Sistema computacional
- Herramientas software
 - Valgrind

Valgrind

- Software para depurar problemas de memoria y de rendimiento en el código fuente.
- Versión actual: 3.23.0
- Instalación (en Ubuntu Linux)

```
sudo apt-get update  
sudo apt-get install -y valgrind
```
- Requisitos:
 - Kernel de Linux 3.0 (o superior)
 - `glibc` 2.5.x (o posterior)

Valgrind

- **Depuración de Errores de Memoria (Memcheck)**
 - Detecta errores de memoria comunes en C/C++
 - Memoria no inicializada.
 - L/E de zonas de memoria que han sido liberadas.
 - L/E fuera de los límites de los bloques de memoria reservados de forma dinámica.
 - Fugas de memoria.
 - No detecta:
 - L/E fuera de rango en arrays reservados de forma estática o en la pila.
 - Ocasionalmente: falsos positivos (errores en código de librerías de sistema)
 - Suprimir con `--gen-suppressions=yes`

Valgrind

- **Depuración de Errores de Memoria**

- Compilar código con opción `-g` (no incluir ninguna del tipo `-Ox`)
- Ejecutar programa con opción `--leak-check=yes`
 - Información extra: `--track-origins=yes`
 - Salida en fichero de texto: `--log-file=<fichero>`

```
valgrind --tool=memcheck --leak-check=yes ./myprog args
```

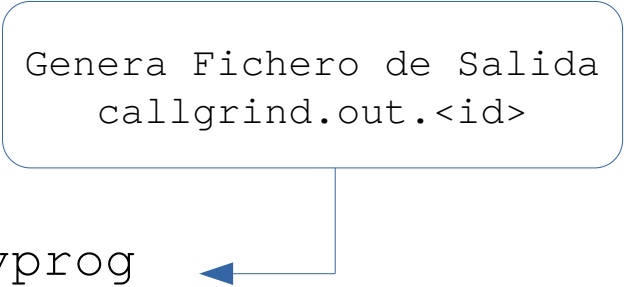
Programa de ejemplo
`prueba.c`

Valgrind

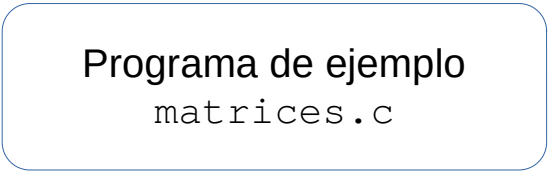
- **Profiling**

- Analiza el tiempo de ejecución empleado por diferentes rutinas del código.
 - Detección de cuellos de botella.
 - Optimizar el rendimiento del programa.
- Compilar código con opción `-g`
- Ejecución: `valgrind --tool=callgrind ./myprog`
- Análisis de información de profiling: `kcachegrind callgrind.out.<id>`
(`apt install kcachegrind`)

Genera Fichero de Salida
`callgrind.out.<id>`



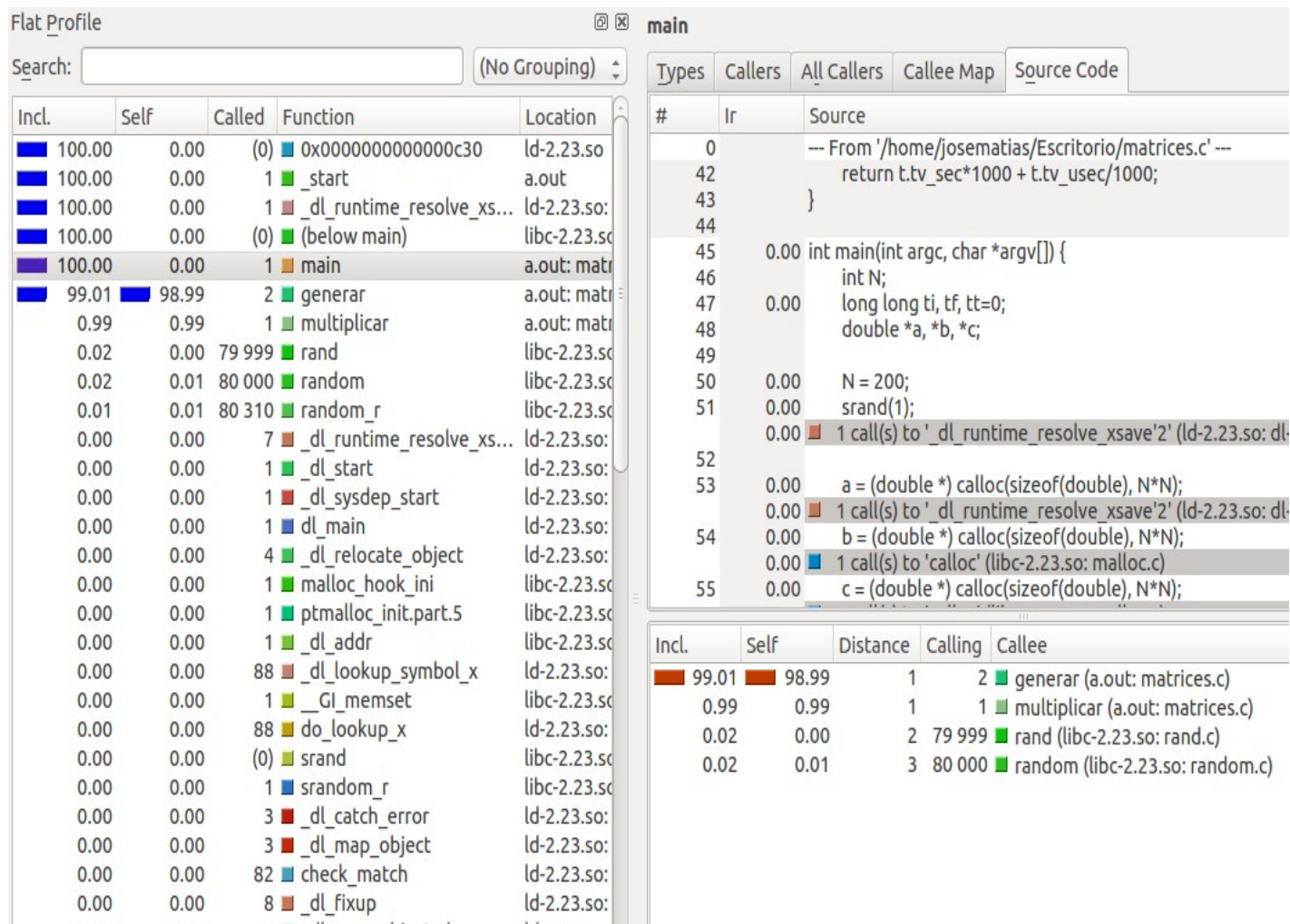
Programa de ejemplo
`matrices.c`



Valgrind

- Profiling
 - kcachegrind

Programa de ejemplo
matrices.c



Valgrind

- Más información: <https://valgrind.org/>
- Manual de Usuario: https://valgrind.org/docs/download_docs.html

Software sesiones posteriores:

OpenMP

GCC versión 4.2.0 o superior ofrece compatibilidad con OpenMP: incorpora sus librerías.

```
gcc --version
```

MPI

Guía instalación en <https://www.open-mpi.org/faq/?category=building>

Actividades Sesión 1 (sin entrega)

- 1 Revisa la Guía Rápida de Usuario de Valgrind disponible en el Aula Virtual en */Prácticas/Práctica_0/ Sesión_1/Herramientas/Depuracion+Profiling/Valgrind.pdf*.

Ejecuta los programas de la carpeta */Codigos* a los que referencia la Guía, y comprueba la utilidad de Valgrind para la depuración de errores de memoria y rendimiento.

- 2 Experimenta con el paralelismo con OpenMP y MPI en el sistema computacional, compilando y ejecutando los códigos disponibles en el directorio indicado en la transparencia 4. Para ello, es suficiente con ejecutar el script *run.sh* en cada caso.

Se trata de un algoritmo simple de multiplicación de matrices en sus versiones secuencial y paralelas en memoria compartida y paso de mensajes.

Puedes comparar los tiempos de las distintas ejecuciones y comprobar si son razonables teniendo en cuenta el efecto del paralelismo.