Trabajo Teórico: Herramientas de Análisis de Rendimiento

Pablo Rodríguez Solera / Juan Mena Patón

Computadores Avanzados

➔ Prof. Serafín Benito Santos.

➔ Escuela Superior de Informática.

➔ Universidad de Castilla – La Mancha

Índice:

Trabájo Teórico: Herramientas de Análisis de Rendimiento

1. Introducción:

El análisis del rendimiento de una aplicación paralela puede resultar muy interesante debido a toda la información que podemos obetener del mismo.

Con la información podemos identificar cuellos de botella dentro de nuestra aplicación, entender que uso del hardware disponible hace nuestra aplicación, los diferentes consumos que conlleva etc. Y en consecuencia, mejorar así o reparar los posibles fallos o puntos de mejora que identifiquemos.

Como primer caso de estudio, vamos a utilizar la herramienta Intel Vtune Profiler, anteriormente conocida como Intel Vtune Amplifier. Esta herramienta nos va a permitir analizar aplicaciones con componente paralelo tanto MPI como OpenMP y visualizar diferentes datos de la ejecución.

2. Intel VTune Profiler:

Esta herramienta está diseñada por intel y nos permite realizar muchas funciones en relación con lo explicado más arriba.

2.1 Instalación:

El primer paso es instalarla. Está disponible para la mayoría de sistemas operativos. En nuestro caso hemos decidido instalarla en un Linux nativo de 64 bits (debian).

Para poder descargarla hace falta una licencia, nosotros hemos obtenido la licencia de estudiante y desde la página de intel, solo hace falta registrarse y te permite descargarla según tu distribución.

Una vez descargada, simplemente descomprimimos el fichero y nos encontramos 2 archivos de instalación, una versión para consola de comandos y otra con interfaz gráfica.

Nosotros hemos escogido la opción con interfaz gráfica debido a que es un proceso más fácil e intuitivo.

Figura 1: Contenido de la carpeta al descomprimir el fichero.

Para realizar la instalación es recomendable realizarla con permisos de superusuario, ya que aunque nos deja hacerlo como usuario normal nos avisa de que es recomendable realizarlo como superusuario para evitar fallos por permisos en alguna creación de ficheros o carpetas.

Al ejecutar el instalador se nos abre una ventana clásica de instalación guiada por pasos:

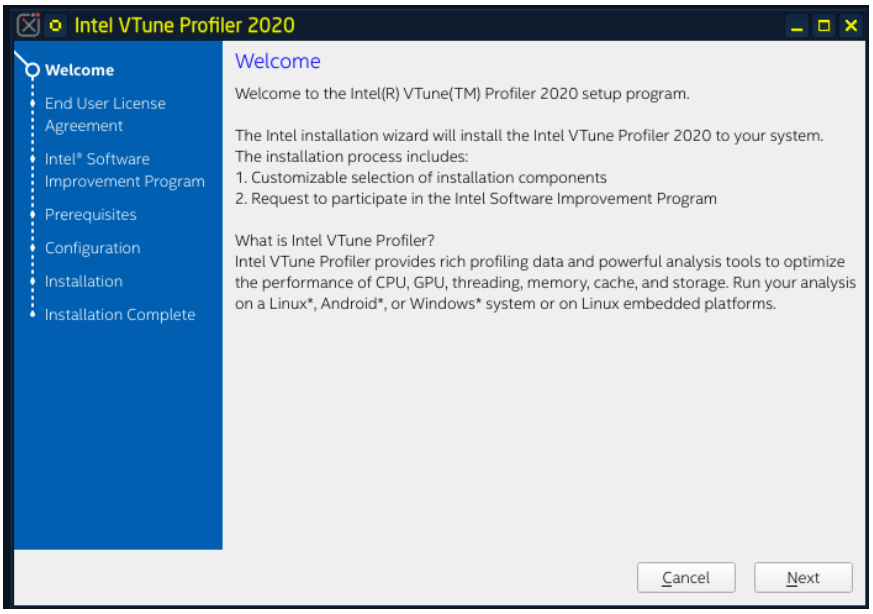

Figura 2: Ventana de instalación Vtune.

Vamos siguiendo los pasos que nos va describiendo el instalador, como elegir la carpeta de destino o informarnos de lo que se va a instalar.

Antes de proceder con la instalación tenemos que aceptar los terminos de usuario y si queremos dar información a intel sobre nuestras pruebas.

Después se lleva a cabo la instalación y al finalizar si todo ha ido como debería, nos informa de que ya está instalada y podemos comenzar a utilizarla.

2.2 Archivos necesarios

Para poder realizar el análisis de nuestros programas con VTune no es necesaria ninguna configuración especial.

Por ejemplo en el caso de un programa con paralelismo OpenMP, simplemente necesitamos el archivo fuente con el código del programa y el programa compilado con la opción de depuración, es decir con el flag -g (a parte de los necesarios para compilar el programa). De esta forma un ejemplo sería:

➔ `gcc prueba.c –o prueba –fopenmp –g`

2.3 En la aplicación

Si estamos en uns distribución linux, la instalación por defecto se realiza en la carpeta opt de la raíz. La instalación incluye un archivo .desktop para poder incluirla en nuestro registro de aplicaciones, pero no lo hace por defecto por lo que para iniciarla debemos entrar en la carpeta de instalación. Entramos y nos encontramos con el fichero ejecutable dentro de la carpeta de 64 o 32 bits según la arquitectura de nuestro equipo.

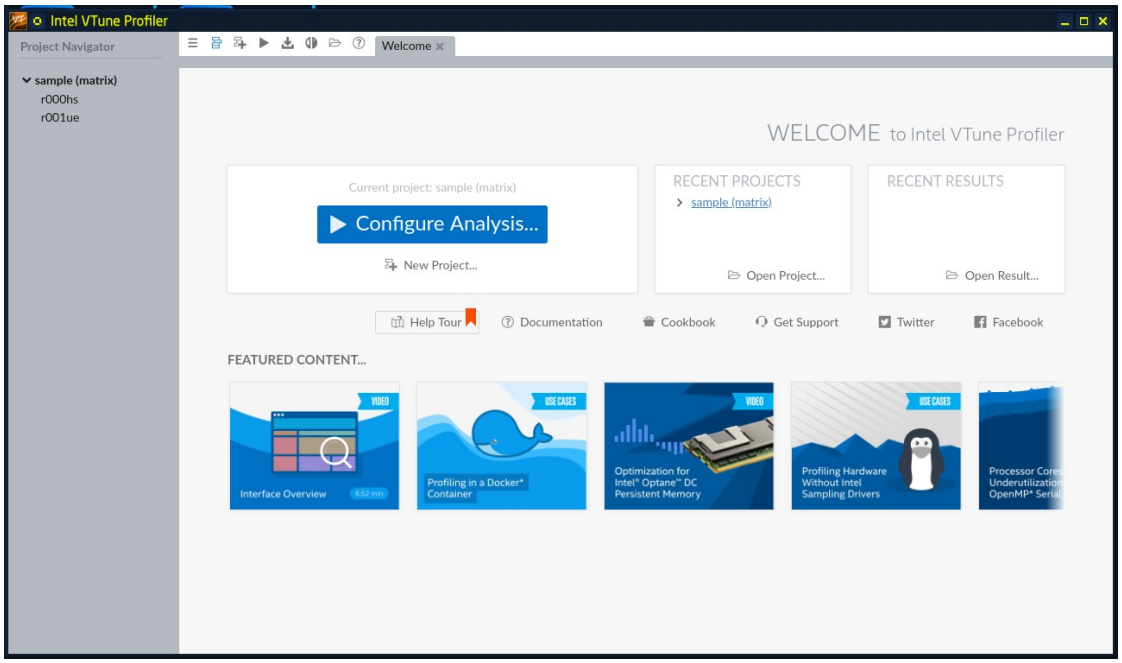

Figura 3: Pantalla de bienvenida de VTune

Iniciamos el programa y nos encontramos con la pantalla de bienvenida.

2.3.1 Crear un proyecto

Crear un proyecto con VTune es fácil con la interfaz gráfica. Dentro de la ventana de bienvenida nos encontramos con el botón de crear un nuevo proyecto, al pulsarlo, se abre una ventana como la siguiente:

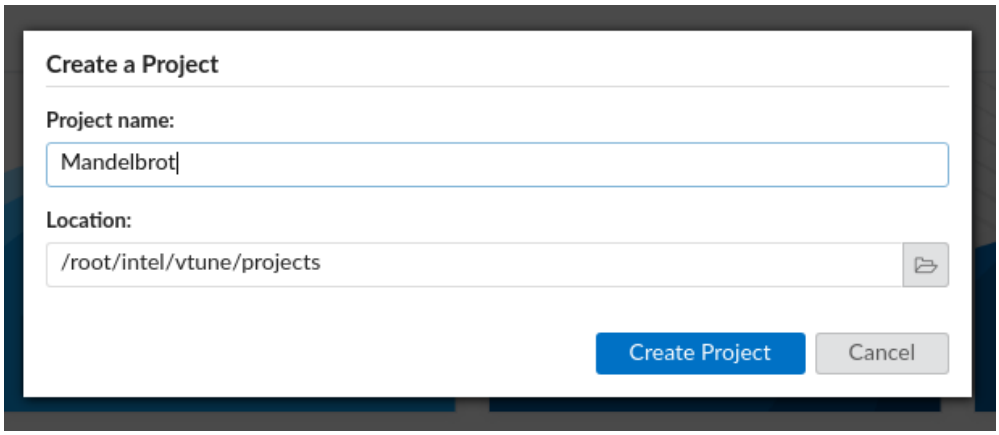

Figura 4: Creación de un proyecto

Ponemos nombre a nuestro proyecto y lo creamos.

Una vez se crea se nos muestra una pantalla con información del proyecto y algunas opciones y configuraciones que podemos realizar sobre el mismo.

Lo primero que tenemos que hacer es añadir el fichero compilado de la aplicación que queremos analizar.

Para ello seleccionamos el fichero en la sección "Application".

Si nuestro archivo necesita algún parámetro especial en su ejecución debemos añadirlo en la sección "Application parameters". En nuestro caso solo es un fichero con datos de entrada que necesita nuestra aplicación.

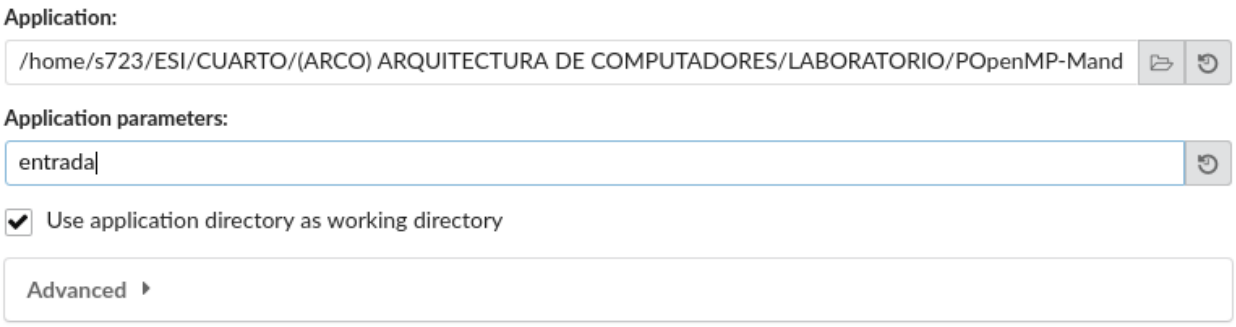

Figura 5: Añadir la aplicación al proyecto.

Al VTune Profiler le hace falta también el archivo con el código del programa sin compilar para poder informarnos sobre las funciones dentro del mismo. Para dárselo tenemos que irnos a la opción "Search Sources/Binaries" que se encuentra abajo junto al botón de iniciar.

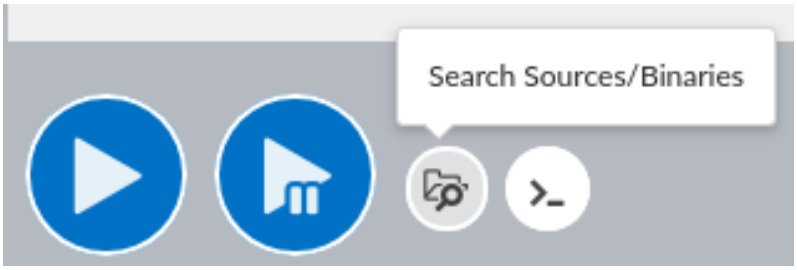

Figura 6: Botón de añadir binarios.

Aquí podemos ver los botones del programa que nos permiten ejecutar un análisis, además, esta sección nos permite añadir fuentes o binarios, en nuestro caso queremos añadir los fuentes, así que entramos en la sección "Sources" y pulsamos a la derecha el botón de buscar carpeta.

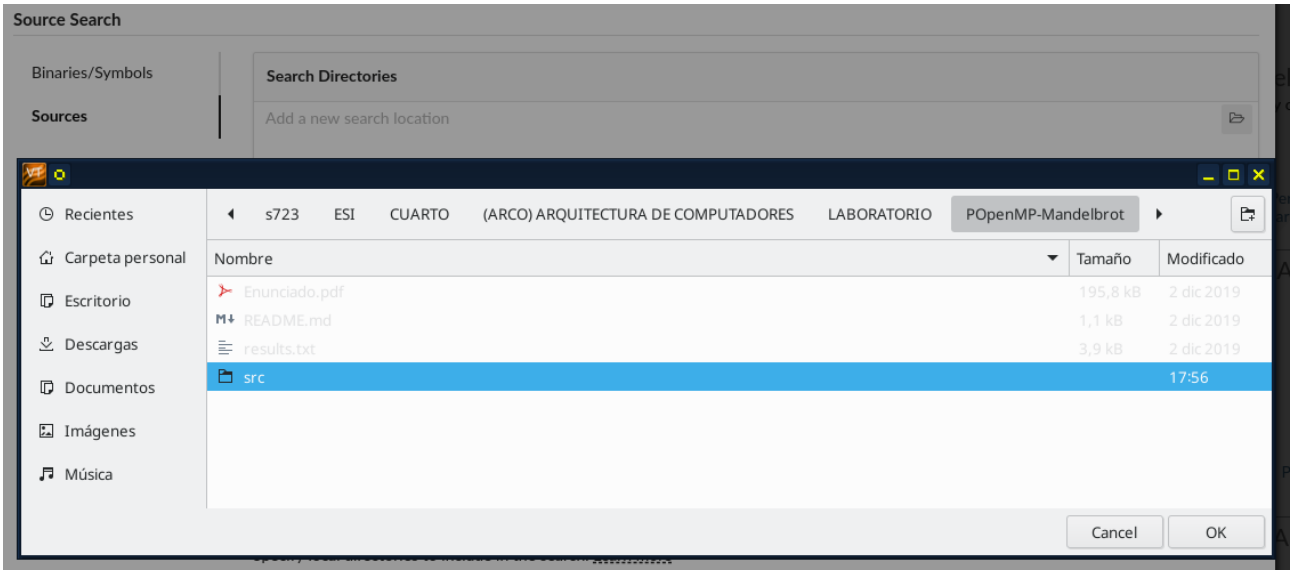

Figura 7: Añadimos la carpeta con los fuentes del programa.

Aquí buscamos la carpeta que contiene los archivos de nuestro programa y la seleccionamos. VTune se encarga solo de buscar que archivo corresponde a nuestro ejecutable, así que seleccionamos la carpeta src entera.

Una vez tenemos todo preparado, solo nos queda ejecutar el análisis para empezar a ver los resultados y la información sobre nuestra aplicación.

2.3.2 Ejecutando un análisis

Para empezar el análisis, simplemente tenemos que pulsar el botón de "Play" azul que tenemos abajo a la izquierda. Una vez lo pulsemos el programa empezará a ejecutar nuestra aplicación.

Una vez ejecutemos, podremos ver como se va ejecutando el análisis y las fases que va recorriendo hasta finalizar.

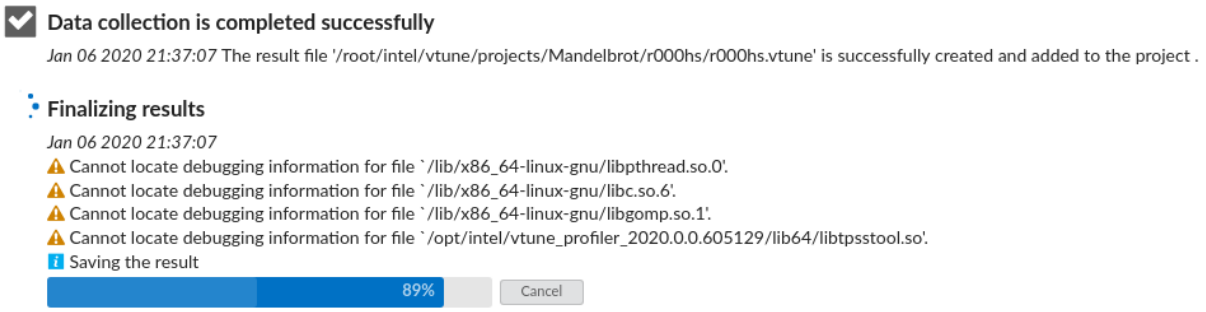

Figura 8: Ejecutando el análisis.

3. OpenMP - Mandelbrot

Para realizar los análisis de rendimiento, vamos a utilizar como aplicación la correspondiente a la práctica final de Arquitectura de Computadores de este curso, que se corresponde con una aplicación que busca números complejos que correspondan a los fractales dentro del conjunto de Mandelbrot.

Este programa está escrito en C y tenemos varias versiones que vamos a analizar.

En primer lugar vamos a realizar un análisis de la versión con el programa secuencial y después con las versiones del programa paralelo.

3.1 Explicación del programa

3.1.1 Funciones

El programa como ya hemos explicado busca números complejos dentro del conjunt Mandelbrot y después imprime una serie de imagenes del conjunto.

Vamos a realizar la explicación de las funciones del programa así como de las comunicaciones que se producen entre los procesos en la versión paralela.

- `main()`: La función principal del programa, se encarga de abrir el fichero con los límites de cada una de las imagenes en las cuales se va a realizar la búsqueda del conjunto y que queremos generar.
  - Lee el tipo de imagen, si es circular o rectangular.
  - Calcula la anchura y la altura de la imagen en píxeles.
  - Calcula la coordenada (x,y) y decide si se incluye en el conjunto mandelbrot.
  - Determina el color del pixel y lo imprime en el archivo de imagen que ha creado.
- `mandel_val()`: Es la función que se determina si el número complejo (x,y) a comprobar tiene posibilidades de pertenecer al conjunto de Mandelbrot o rotundamente no pertenece al mismo. Esta función es en la que se busca mejorar el rendimiento en la versión paralela del programa.

3.1.2 Versión paralela

En la versión paralela, tenemos que crear una serie de variables tanto públicas como privadas para cada proceso, además, utilizamos las directivas correspondientes de OpenMP para paralelizar la sección de código que necesitamos.

En este caso se paraleliza la directiva for que va recorriendo todos los pixeles y comprobando que números del plano (x,y) corresponden al conjunto.

En esta versión, simplemente hacemos uso de la directiva parallel, separamos las regiones de la imagen que le tocan a cada proceso y le indicamos que las variables x e y correspondientes a los pixeles son privadas.

En este caso la comunicación entre procesos es inexistente, puesto que el proceso primitivo calcula que región le toca a cada proceso hijos y estos solo modifican información dentro de su región por lo que no se comunican entre ellos.

3.1.3 Versión paralela mejorada

Esta versión también tiene paralelismo, pero en este caso hacemos uso de una directiva de OpenMP específica para paralelizar un for y así no tener que realizar los cálculos de regiones para cada proceso de manera manual, por lo que no hacen falta las variables de antes y se simplifica algo el código.

Igualmente las variables x e y siguen siendo privadas y la comunicación entre procesos inexistente. Además incluimos la opción del scheduling, para poder realizar distintas configuraciones del mismo y ver los cambios en el rendimiento.