Titulo del trabajo ☣

Diego Alejandro Londoño Jiménez
diegoa.londono@udea.edu.co

Despartamento de Ingeniería Electrónica y Telecomunicaciones
Universidad de Antioquia
Medellín
Mayo de 2022

Índice

1. Introducción

2. Preguntas orientadoras

2.1. En sus propias palabras defina que es *profiling*

Análisis detallado del código que se tiene para saber dónde se está consumiendo más ciclos de reloj el procesador, teniedo claro lo anterior se puede proceder a remplazar la función o las bibliotecas en caso de que se pueda y de esta manera poder optimizar el código.

2.2. ¿Qué papel desempeña la creación de los CFG y los CallGraph en el "profiling" de un determinado código?

- **CFG:** Es un modelo abstracto del código fuente sin importar el lenguaje del código fuente, modela la operación de datos (aritméticas, matriciales) y operación de control(condicionales). Permite ver como evolucionana y cuál es la dependencia que hay de los datos, una de las ventajas de ver esa dependencia y de poder contar varios *core* en el procesador, es que se pueden usar los diferentes *core* para las operaciones que no tengan dependencia.
- **CallGraph:** Muestra cuál es el ordenamiento de llamadas a funciones, qué función llama a cuál función,tanto funciones propias, como funciones de bibliotecas que hagan parte del lenguaje.

2.3. Consulte y describa brevemente en qué consisten las herramientas Kcachegrind y Valgrind.

> "**Valgrind** es un programa flexible para depurar y hacer análisis detallado al código ejecutable de Linux. Consta de un núcleo, que proporciona una CPU sintética en software, y una serie de herramientas de depuración y profiling. La arquitectura es modular, de modo que se pueden crear nuevas herramientas fácilmente y sin alterar la estructura existente." [1] En pocas palabras, es una herramienta que me permite hacer un profiling detallado del código y a encontrar dónde se está consumiendo más instrucciones de reloj el código

KCachegrind, es una herramienta que permite visualizar los datos de los profiling realizados al código.

2.4. En el contexto de profiling, ¿en qué consisten las fugas de memoria (*memory leaks*)

> "Una fuga de memoria (más conocido por el término inglés memory leak) es un error de software que ocurre cuando un bloque de memoria reservada no es liberada en un programa de computación. Comúnmente ocurre porque se pierden todas las referencias a esa área de memoria antes de haberse liberado.

Dependiendo de la cantidad de memoria perdida y el tiempo que el programa siga en ejecución, este problema puede llevar al agotamiento de la memoria disponible en la computadora. Este problema se da principalmente en aquellos lenguajes de programación en los que el manejo de memoria es manual (C o C++ principalmente), y por lo tanto es el programador el que debe saber en qué momento exacto puede liberar la memoria. Otros lenguajes utilizan un recolector de basura o conteo de referencias que automáticamente efectúa esta liberación. Sin embargo todavía es posible la existencia de fugas en estos lenguajes si el programa acumula referencias a objetos, impidiendo así que el recolector llegue a considerarlos en desuso." [2]

2.5. ¿Por qué cree usted que las fugas de memoria se deben evitar en los Sistemas de Tiempo Real?

Si se llega al punto de agotar la memoria disponible de la computadora y dependiendo del proceso que se esté ejecutando puede traer consecuencias muy delicadas, hasta llegar al punto de tener perdidas humanas, por tal motivo el programador debe de garantizar siempre que esto no suceda,

3. Profiling de un programa de prueba

3.1. Compilación

3.1.1. Qué hace el programa *prueba.c*

Se importa la biblioteca $< valgrind/callgrind.h >$, es la que se encarga de ejecutar los comandos $CALLGRIND\_START\_INSTRUMENTATION$ y $CALLGRIND\_STOP\_INSTRUMENTATION$. El programa consta de tres funciones enteras que retornan las operaciones correspondientes. la función int add() retorna la suma de la variable val de cero hasta 799, la función int mult() retorna la multiplicación de la variable val de cero hasta 799, la función foo() retorna el valor de la función add() dos veces más la función mult()

3.1.2. ¿Cuál es el papel que desempeña el $\#include < valgrind/callgrind.h >$, así como las instrucciones de preprocesamiento $CALLGRIND\_START\_INSTRUMENTATION$ y $CALLGRIND\_STOP\_INSTRUMENTATION$?

$\#include < valgrind/callgrind.h >$ se encarga de traer del programa valgrind la biblioteca *callgrind.h* la cual ejecuta las instrucciones $CALLGRIND\_START\_INSTRUMENTATION$ y $CALLGRIND\_STOP\_INSTRUMENTATION$, los macros anteriores se ubican en la

posición del código en la que debe comenzar y finalizar la elaboración del profiling y se usa para hacer el profiling especialmente a una parte del código.

3.2. Generación del Callgraph del programa "Prueba"

3.2.1. Consulte e indique qué hacen las opciones $--dump-instr=yes$ y $--instr-atstart=no$

$--dump-instr=yes$: Permite ver el código ensamblador, los datos del perfil resultante sólo se pueden verse con KCachegrind. $--instr-atstart=no$ Incializa Callgrin con el modo de instrumentación desactivado

3.2.2. En el entorno gráfico del Kcachegrind observe el contenido de la ventana inferior llamada "Call Graph". Analice este grafo presentado y interprete los datos numéricos presentados sobre dicho grafo.

Los valores hacen referencia a la cantidad de ciclos de reloj que se demora cada bloque en realizar lo que le corresponde ejecutar.

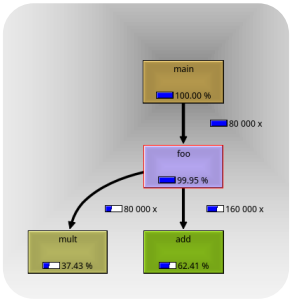

Figura 1: CallGraph

3.2.3. Análisis de Callee Map, Source Code, Machine Code, Callees, All Callers

Callee Map: Estando en esta pestaña, el usuario se puede desplazar en la lista de Flat Profile y se puede ver en Callee Map las funciones que son llamados y cuál función contiene a cuál función a medida que se desplaza en Flat Profile. **Source Code:**Muestra la función que se le esté haciendo el profiling, dicha función se encarga de llamar las otras dos funciones, por lo tanto es la que consume más ciclos de reloj en ejecución diferente al main **Callees: All Callers:**

4. Conclusiones

- xxxxxxx
- yyyyyyy
- zzzzzzz

Referencias

[1] Linux, "Valgrind documentation," 2022. [Online]. Available: https://valgrind.org/docs/manual/valgrind_manual.pdf

[2] Wikipedia, "Fuga de memoria," 14 nov 2020 a las 17:58. [Online]. Available: https://es.wikipedia.org/wiki/Fuga_de_memoria