INSTITUTO TECNOLÓGICO DE NUEVO LEÓN

Ingeniería en sistemas computacionales

Lenguaje y Autómatas 2

Resumen Unidad 3

Catedrático

Juan Pablo Rosas Baldazo

Alumno

Axel Johnary Liñan Estrada

15/04/2018

Introducción

La optimización consiste en mejorar el código intermedio de modo que quede un código maquina más rápido de ejecutar. La optimización es un proceso que tiene a minimizar o maximizar alguna variable de rendimiento, generalmente tiempo, espacio, procesador, etc.

Capítulo 1: Tipos de optimización

Dentro de los tipos de optimización se derivan los tipos de optimización local, optimización de ciclo, optimización global y optimización de mirilla.

Sección 1.1: Locales

La optimización local se realiza sobre módulos del programa. En la mayoría de las ocasiones a través de funciones, métodos, procedimientos, clases, etc. La característica de las optimizaciones locales es que solo se ven reflejados en dichas secciones.

La optimización local sirve cuando un bloque de programa o sección es crítico por ejemplo: E/S, la concurrencia, la rapidez y confiabilidad de un conjunto de instrucciones. Como el espacio de soluciones es más pequeño la optimización local es más rápida.

Sección 1.2: Ciclos

Los ciclos son una de las partes más esenciales en el rendimiento de un programa dado que realizan acciones repetitivas, y si dichas acciones están mal realizadas, el problema se hace N veces más grandes. La mayoría de las optimizaciones sobre ciclos tratan de encontrar elementos que no deben repetirse en un ciclo.

Sea el ejemplo:

```
while(a == b) {
int c = a;
c = 5;
…;
}
```

En este caso es mejor pasar el Int c =a; fuera del ciclo de ser posible.

El problema de la optimización en ciclos y en general radica es que muy difícil saber el uso exacto de algunas instrucciones. Así que no todo código de proceso puede ser optimizado. Otros uso de la optimización pueden ser el mejoramiento de consultas en SQL o en aplicaciones remotas (sockets, E/S, etc.)

Sección 1.3: Globales

La optimización global se da con respecto a todo el código. Este tipo de optimización es más lenta pero mejora el desempeño general de todo programa. Las optimizaciones globales pueden depender de la arquitectura de la máquina.

En algunos casos es mejor mantener variables globales para agilizar los procesos (el proceso de declarar variables y eliminarlas toma su tiempo) pero consume más memoria. Algunas

optimizaciones incluyen utilizar como variables registros del CPU, utilizar instrucciones en ensamblador.

Sección 1.4: De mirilla

La optimización de mirilla trata de estructurar de manera eficiente el flujo del programa, sobre todo en instrucciones de bifurcación como son las decisiones, ciclos y saltos de rutinas.

La idea es tener los saltos lo más cerca de las llamadas, siendo el salto lo más pequeño posible.

Capítulo 2: Costos

Los costos son el factor más importante a tomar en cuenta a la hora de optimizar ya que en ocasiones la mejora obtenida puede verse no reflejada en el programa final pero si ser perjudicial para el equipo de desarrollo. La optimización de una pequeña mejora tal vez tenga una pequeña ganancia en tiempo o en espacio pero sale muy costosa en tiempo en generarla.

Pero en cambio si esa optimización se hace por ejemplo en un ciclo, la mejora obtenida puede ser N veces mayor por lo cual el costo se minimiza y es benéfico la mejora.

Sección 2.1: Costos de ejecución (Memorias, registros, pilas)

Los costos de ejecución son aquellos que vienen implícitos al ejecutar el programa.

En algunos programas se tiene un mínimo para ejecutar el programa, por lo que el espacio y la velocidad de los microprocesadores son elementos que se deben optimizar para tener un mercado potencial más amplio.

Las aplicaciones multimedia como los videojuegos tienen un costo de ejecución alto por lo cual la optimización de su desempeño es crítico, la gran mayoría de las veces requieren de procesadores rápidos (ej. tarjetas de video) o de mucha memoria. Otro tipo de aplicaciones que deben optimizarse son las aplicaciones para dispositivos móviles.

Los dispositivos móviles tienen recursos más limitados que un dispositivo de cómputo convencional razón por la cual, el mejor uso de memoria y otros recursos de hardware tiene mayor rendimiento. En algunos casos es preferible tener la lógica del negocio más fuerte en otros dispositivos y hacer uso de arquitecturas descentralizadas como cliente/servidor o P2P.

Sección 2.2: Criterios para mejorar el código

La mejor manera de optimizar el código es hacer ver a los programadores que optimicen su código desde el inicio, el problema radica en que el costo podría ser muy grande ya que tendría que codificar más y/o hacer su código más legible. Los criterios de optimización siempre están definidos por el compilador.

Muchos de estos criterios pueden modificarse con directivas del compilador desde el código o de manera externa. Este proceso lo realizan algunas herramientas del sistema como los ofuscadores para código móvil y código para dispositivos móviles.

Sección 2.3: Herramientas para el análisis del flujo de datos

Existen algunas herramientas que permiten el análisis de los flujos de datos, entre ellas tenemos los depuradores y desensambladores. La optimización al igual que la programación es un arte y no se ha podido sistematizar del todo.

Conclusiones

La optimización es importante a la hora de diseñar o programar un código, porque hay que reducir los tiempos y recursos necesarios para mejorar los requisitos mínimos necesarios para ejecutar el programa final y así ser mejor que la competencia ya que sería más accesible.

Conceptos

Rendimiento: Medida o cuantificación de la velocidad/resultado con que se realiza una tarea o proceso. En una computadora, su rendimiento no depende sólo del microprocesador como suele pensarse, sino de la suma de sus componentes como la memoria, el bus, los diversos dispositivos, etc. y su software.

Concurrencia: es una propiedad de los sistemas en la cual los procesos de un cómputo se hacen simultáneamente, y pueden interactuar entre ellos.

Saltos: Los términos de salto o rama suelen utilizarse para referirse a programas escritos en lenguaje máquina o en lenguaje ensamblador; en los lenguajes de alto nivel, los saltos normalmente toman la forma de sentencias condicionales, llamadas a subrutinas o sentencias GOTO.

Llamadas: es el mecanismo usado por una aplicación para solicitar un servicio al sistema operativo.

Ofuscación: En computación, la ofuscación se refiere al acto deliberado de realizar un cambio no destructivo, ya sea en el código fuente de un programa informático o código máquina cuando el programa está en forma compilada o binaria, con el fin de que no sea fácil de entender o leer. El código ofuscado es aquel código que, aunque se tiene el código fuente, ha sido enrevesado específicamente para ocultar su funcionalidad.

Depuración: La depuración de programas es el proceso de identificar y corregir errores de programación.

Desensamblador: Un desensamblador es un programa de computador que traduce el lenguaje de máquina a lenguaje ensamblador, la operación inversa de la que hace el ensamblador.

Bibliografía o Referencias

Lenguajes y Autómatas II optimización, descrita en http://www.academia.edu/29416690/lenguajes_y_automatas_ii_optimizacion

Unidad 3: Optimización, descrita en http://itpn.mx/recursosisc/7semestre/leguajesyautomatas2/Unidad%20III.pdf

Tipos de optimización, descrita en http://noeliy22.blogspot.mx/2013/11/tipos-de-optimizacion.html

Ofuscación, descrita en https://es.wikipedia.org/wiki/Ofuscaci%C3%B3n

Rendimiento, descrita en http://www.alegsa.com.ar/Dic/rendimiento.php

Concurrencia (informática), descrita en https://es.wikipedia.org/wiki/Concurrencia_(inform%C3%A1tica)

Llamada al sistema, descrita en https://es.wikipedia.org/wiki/Llamada_al_sistema

Salto (informática), descrita en https://es.wikipedia.org/wiki/Salto_(inform%C3%A1tica)

Depuración en programas, descrita en https://es.wikipedia.org/wiki/Depuraci%C3%B3n_de_programas

Desensamblador, descrita en https://es.wikipedia.org/wiki/Desensamblador

Reporte

La optimización es un proceso para mejorar la ejecución de un programa, puede ser mejorando los tiempos de ejecución o reducir los recursos necesarios para ejecutarse. Estas mejoras en un programa pequeño tal vez sean de unos cuantos milisegundos de diferencia, pero en un sistema mucho mayor si se reducen grandes tamaños de tiempo y/o recursos.

Existen 4 tipos de optimización: locales, ciclos, globales y de mirilla. Las optimizaciones locales consisten en los métodos, funciones y clases. Los ciclos consisten en ciclos como el while, do while, for. Los ciclos globales consisten en el manejo de variables globales y registros en ensamblador. Y por último, las optimizaciones de mirilla, que consisten en acortar las distancias de los saltos en los algoritmos del código.

Los costos para optimizar un código son altos porque hay que dedicar mucho tiempo en analizar todo el código programado para mejorarlo, por lo que se recomienda aplicar técnicas de optimización desde el inicio.