INSTITUTO TECNOLÓGICO DE NUEVO LEÓN

Ing. Sistemas computacionales

"Lenguajes Autómatas 2"

Profesor: Juan Pablo Rosas Baldazo

Tarea: Reporte de unidad

Unidad 3

Presenta.

Mario Humberto Uriegas de León

No. De control: 14480514

Cd. Guadalupe; Nuevo León; a 15 de Abril de 2018

Introducción

En el siguiente reporte se verá un resumen de los tipos de optimización ya sean locales, de ciclos, globales o de mirilla al igual que se observaran los costos, manejando los costos de ejecución sus criterios para mejorar el código o las herramientas para el análisis del flujo de datos.

Capítulo 1: Tipos de optimización

Las optimizaciones pueden realizarse de diferentes formas, estas se realizan en base a el alcance ofrecido por el compilador.
La optimización va a depender del lenguaje de programación y es directamente proporcional al tiempo de compilación; es decir, entre más optimización mayor tiempo de compilación.

Como el tiempo de optimización es gran consumidor de tiempo (dado que tiene que recorrer todo el árbol de posibles soluciones para el proceso de optimización)la optimización se deja hasta la fase de prueba final. Algunos editores ofrecen una versión de depuración y otra de entrega o final.

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 sólo se ven reflejados en dichas secciones. Esta sirve cuando un bloque de programa o sección es crítico por ejemplo; la 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.
El problema de la optimización en ciclos y en general radica en que es muy difícil saber el uso exacto de algunas instrucciones. Así que no todo código de proceso puede ser optimizado. Otro uso de la optimización puede ser en 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.

Grafo del flujo de ejecución: antes de realizar una optimización global es necesario crear el grafo de flujo de ejecución. El grafo de flujo de ejecución representa todos los caminos posibles de ejecución del programa. La optimización global a partir del análisis del grafo del flujo de ejecución permite:
-Una propagación de constantes fuera del bloque básico.
-Eliminación del código no utilizado
-Una mejor asignación de los registros.

Sección 1.4: **De mirilla**

La optimización de mirilla trata de estructurar de manera eficiente el flujo del programa, sobre todo en instrucciones de **bifurcaciones** 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. Algunas ideas básicas son:

-Se recorre el código buscando combinaciones de instrucciones que pueden ser reemplazadas por otras equivalentes más eficientes.

-Se utiliza una ventana de "n" instrucciones y un conjunto de patrones de transformación (patrón, secuencias, etc.).

-Las nuevas instrucciones son reconsideradas para las futuras optimizaciones.

Algunos ejemplos con la eliminación de cargas innecesarias la reducción de potencia y la eliminación de cadenas de saltos.

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 sin 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 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 benéfica la mejora.

Sección 2.1: **Costo de ejecución (memoria, 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 crítica, la gran mayoría de las veces requieren de procesadores rápidos (e.g. 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 lo 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

En este trabajo se observaron los tipos de optimización como son la optimización local que es la que se realiza sobre módulos del programa, en la optimización de ciclos o bucles se mostró que son esenciales en el rendimiento del programa ya que realizan acciones repetitivas, en cuanto a la optimización global puede depender de la arquitectura de la máquina, es as lenta pero mejora el desempeño en general, la optimización de mirilla trata de estructurar el flujo del programa de manera eficiente, en este trabajo observamos los costos que son el factor mas importante a tomar en cuenta cuando se va a optimizar, en esto se maneja costo de ejecución como son de memoria, registros o pilas, los criterios para mejorar un código y las herramientas necesarias para el análisis del flujo de datos.

Conceptos

Concurrencia: es una propiedad de los sistemas en la cual los procesos de un cómputo se hacen simultáneamente, y pueden interactuar entre ellos. También se puede interpretar que distintas personas o maquinas lleguen al mismo resultado o conclusión.

Bifurcación: División de una cosa en 2 ramas, brazos o puntas.

P2P: Es una red de ordenadores en la que todos o algunos aspectos funcionan sin clientes ni servidores fijos, sino una serie de nodos que se comportan como iguales entre sí.

Depurador: Es un programa usado para probar y depurar (eliminar) los errores de otros programas (el programa "objetivo").

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

[1] Juan Carlos Santiago. (2016). Optimización Global. 14/04/19, de Blogger.com Sitio web: http://juancarlossant.blogspot.mx/2016/11/optimizacion-global.html

[2] Gabriela Fernández Espinoza. (2013). Tipos de optimización. 15/04/18, de Blogger.com Sitio web: http://gaferz.blogspot.mx/2013/11/tipos-de-optimizacion.html

[3] Juan Carlos Olivares Rojas. (-). Optimización. 15/04/2018, de IT Morelia Sitio web: http://dsc.itmorelia.edu.mx/~jcolivares/courses/ps207a/ps2_u7.pdf

[4] ITPN. (2012). Optimización. 15/04/18, de ITPM.mx Sitio web: http://itpn.mx/recursosisc/7semestre/leguajesyautomatas2/Unidad%20III.pdf

Reporte de aprendizaje

Bueno según lo que entendí la optimización busca que un programa o el sistema o un proceso tengan el mejor resultado posible, la optimización tiene varias formas de lograrse y estas a su vez se van a realizar en base al alcance que determine el compilador o dependerá del lenguaje elegido, entre mejor quede un programa ósea mejor optimizado este tardara más tiempo en compilarse, al parecer la optimización local es más rápida porque tiene un espacio de soluciones más pequeño, los bucles son muy útiles para hacer acciones que se tienen que repetir mucho pero se debe tener en cuenta que si están mal hechas se harán problemas más grandes, de la optimización global solo entendí que dependía de tu propio equipo y que era la más lenta me imagino que por lo mismo, pero como es global implica mejorar todo el programa, en cuanto a la de mirilla entendí que trata de tener una mejor estructura cuando el programa se bifurca (se divide) por así decirlo en sus instrucciones, los costos como es lógico se toman en cuenta para la optimización por los problemas que pueden causar al equipo de desarrollo cuando una mejora no se note en el proyecto al final pero si el costo, en cuanto los costó de ejecución ya vienen por defecto al ejecutar el programa, al parecer los criterios para optimizar siempre estarán definidos por el compilador y la mejor manerade optimizar el código es haciendo que los programadores lo optimicen desde el inicio pero esto puede ser muy costoso por la cantidad de código en cuanto a las herramientas el depurador es un programa que prueba y elimina los errores de otros programas y el desensamblador hace lo inverso que el ensamblador.