INSTITUTO TECNOLÓGICO DE NUEVO LEÓN

Ingeniería En Sistemas Computacionales

Lenguajes y Autómatas II

U3

Tema: Optimización

Proyecto 3 Resumen

Profesor.
Juan Pablo Rosas Baldazo

Presenta.

Carlos Enrique Bernal Araujo

15480004

Índice

Introducción

En este documento se hablará sobre que es la optimización, para que sirve, los diferentes tipos de optimización que hay. También se menciona sobre los costos que esto implica, como se puede mejorar el código y unas herramientas para el óptimo entendimiento del flujo de datos.

1. Tipos de Optimización

La optimización busca mejorar la forma en que un programa utiliza los recursos. Las optimizaciones se realizan en base al 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.

La optimización es un proceso que tiende a minimizar o maximizar alguna variable de rendimiento, generalmente tiempo, espacio, procesador, etc.

Desafortunadamente no existen optimizador que hagan un programa más rápido y que ocupe menor espacio.

La optimización se realiza reestructurando el código de tal forma que el nuevo código generado tenga mayores beneficios. La mayoría de los compiladores tienen una optimización baja, se necesita de compiladores especiales para realmente optimizar el código. Cada optimización está basada en una función de coste y en una transformación que preserve el significado del programa.

Mediante la función de coste queremos evaluar la mejora que hemos obtenido con esa optimización y si compensa con el esfuerzo que el compilador realiza para poder llevarla a cabo. Los criterios más comunes que se suelen emplear son el ahorro en el tamaño del código, la reducción del tiempo de ejecución y la mejora de las necesidades del espacio para los datos del programa.

En cuanto a preservar el significado del programa, es lógico que no tendría sentido realizar optimizaciones que modificaran el comportamiento del programa. Aunque parezca evidente, puede haber complicadas optimizaciones que fallen en ese aspecto.

Existen diversas técnicas de optimización que se aplican al código generado para un programa sencillo. Por programa sencillo entendemos aquel que se reduce a un sólo procedimiento o subrutina. Las técnicas de optimización a través de varios procedimientos se reducen a aplicar las vistas aquí a cada uno de los procedimientos y después realizar un análisis interprocedural.

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. La optimización local 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.

Ejemplos:

1- Pre calcular expresiones constantes (con constantes o variables cuyo valor no cambia).

3! i = 5

j = 4

f = j + 2.5

!

j = 4

f = 6.5

2- Reutilización de expresiones comunes

a = b + c

d = a - d

e = b + c

f = a - d

!

a = b + c

d = a - d

e = a

f = a – d

1.2 Bucles

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.

```
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 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 el mejoramiento de consultas en SQL o en aplicaciones remotas (sockets, E/S, etc.).

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.

1.4 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.

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, remplazan).

Las nuevas instrucciones son reconsideradas para las futuras optimizaciones.

Algunos ejemplos:

- Eliminación de cargas innecesarias
- Reducción de potencia
- Eliminación de cadenas de saltos

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.

Por ejemplo: for (int i=0; i < 10000; i++); si la ganancia es de 30 ms 300s

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 crítico, 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 la cual, el mejor uso de memoria y otros recursos de hardware tiene mayor rendimiento.

La memoria es uno de los recursos más importantes de la computadora y, en consecuencia, la parte del sistema operativo responsable de tratar con este recurso, el gestor de memoria, es un componente básico del mismo. El gestor de memoria del sistema operativo debe hacer de puente entre requisitos de las aplicaciones y los mecanismos que proporciona el hardware de gestión de memoria.

Los registros del procesador se emplean para controlar instrucciones en ejecución, manejar direccionamiento de memoria y proporcionar capacidad aritmética. Los registros son espacios físicos dentro del microprocesador con capacidad de 4 bits hasta 64 bits dependiendo del microprocesador que se emplee. Los registros son direccionables por medio de una viñeta, que es una dirección de memoria. Los bits, por conveniencia, se numeran de derecha (15,14, 13…3,2,1,0) los registros están divididos en seis grupos los cuales tienen un fin específico y son los siguientes:

- Registros de segmento
- Registros de apuntador de instrucciones
- Registros apuntadores
- Registros de propósito general
- Registros índices
- Registros de banderas

La aparición de lenguajes con estructura de bloque trajo consigo la necesidad de técnicas de alojamiento en memoria más flexibles, que pudieran adaptarse a las demandas de memoria durante la ejecución del programa.

En general los compiladores, la asignación de memoria de variables locales se hace de una forma flexible, atendiendo al hecho de que solamente necesitan memoria asignada desde el momento que comienza la ejecución de la función hasta el momento que esta finaliza. Así, cada vez que comienza la ejecución de un procedimiento (o función) se crea un registro de activación para contener los objetos necesarios para su ejecución, eliminándolo una vez terminada ésta.

Dado que durante la ejecución un programa es habitual que unos procedimientos llamen a otros y estos a otros, sucesivamente, se crea una cadena jerárquica de llamadas a procedimientos. Ya que estos están organizados de manera jerárquica los distintos registros de activación asociados a cada procedimiento (o función) se colocarán en una pila en la que entrarán cuando comience la ejecución del procedimiento y saldrán al terminar el mismo.

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.

2.3 Herramientas para el análisis del flujo de datos

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

Entre las herramientas más importantes están:

- **Depurador:** permite correr otros programas, permitiendo al usuario ejercer cierto control sobre los mismos a medida que estos se ejecutan, y examinar el estado del sistema.
- **Desamblador:** programa de computadora que traduce el lenguaje máquina a lenguaje ensamblador, la operación inversa de la que hace el ensamblador, está dirigido a un lenguaje de alto nivel.
- **Diagrama de flujo:** herramienta de modelización que permite describir, de un sistema, la transformación de entradas y salidas.
- **Diccionario de datos:** listado organizado de todos los elementos de datos que son pertinentes para el sistema, con definiciones precisas y rigurosas que le permite al usuario y al proyectista del sistema tener una misma comprensión de las entradas y salidas.

Conclusión:

Al realizar este trabajo me di cuenta que la optimización juega un papel muy importante en todo, ya que siempre se está buscando una manera de que las cosas se hagan más rápido para que funcionen mejor, claro que aquí se ve enfocado más en lo informático.

Conceptos

- **Compilador:** Un compilador es un programa informático que traduce un programa que ha sido escrito en un lenguaje de programación a un lenguaje común, reúne diversos elementos o fragmentos en una misma unidad.
- **Optimización:** Optimizar quiere decir buscar mejores resultados, más eficacia o mayor eficiencia en el desempeño de alguna tarea. En los ámbitos de la informática y la tecnología, la optimización es el proceso a través del cual se mejora la eficiencia y la rapidez en el funcionamiento de un sistema informático.
- **Concurrencia:** Es una propiedad de los sistemas en la cual los procesos de un cómputo se hacen simultáneamente, y pueden interactuar entre ellos.
- **Ensamblador:** Se refiere a un tipo de programa informático que se encarga de traducir un fichero fuente escrito en un lenguaje ensamblador, a un fichero objeto que contiene código máquina, ejecutable directamente por el microprocesador.
- **Bifurcación:** Es el acto y el resultado de bifurcar o bifurcarse: la división en dos apéndices, partes o ramales. Se llama bifurcación, por lo tanto, al sitio donde algo se bifurca.
- **Ofuscadores:** es un verbo regular que puede referirse a turbar la vista como resultado de un deslumbramiento o de un repentino oscurecimiento, asimismo, puede equivaler a trastornar, perturbar o conturbar las ideas o el pensamiento.

Bibliografía

S.A, S.F, Un. VII. Optimización, https://ingarely.files.wordpress.com/2012/11/unidad-vii.pdf

José M. García Carrazco, S.F, La optimización: una mejora en la ejecución de programas, http://ditec.um.es/~jmgarcia/papers/ensayos.pdf

Gabriela Fernández Espinoza, 13 noviembre 2013, optimización, Mis tareas, http://gaferz.blogspot.mx/2013/11/tipos-de-optimizacion.html

Juan Carlos Olivares Rojas, S.F, Unidad VII Optimización, http://dsc.itmorelia.edu.mx/~jcolivares/courses/ps207a/ps2_u7.pdf

Juan José Sánchez, 19 enero 2014, 3.2.1 Costo de ejecución. (Memoria, registros, pilas), https://prezi.com/m-ft53psccpy/321-costo-de-ejecucion-memoria-registros-pilas/

Claudia Dávila, 21 octubre 2014, 3.2.3 herramientas para el análisis del flujo de datos, https://prezi.com/4dtcp9qnkjbk/323-herramientas-para-el-analisis-del-flujo-de-datos/

Reporte

La optimización es el proceso que ayuda a que cierta parte en el programa aumente en rendimiento, que generalmente lo que se busca con esto es que se hagan las cosas o tareas en el menor tiempo posible, esto en el caso que se quiera maximizar. Pero también se puede minimizar ciertas cosas dependiendo lo que se solicite.

La optimización local en la mayoría de las veces se realiza en las funciones, los métodos y los procedimientos, etc. Pero más que nada se usa cuando alguna parte ya está critica porque esta optimización es la más rápida.

En la optimización de Bucles, los ciclos son muy importantes en el rendimiento de un programa, pero hay que tener en cuenta que si algo está mal en ese ciclo dicha acción se puede hacer N veces más grande, por lo cual aquí se trata de encontrar elementos que no deben repetirse en el ciclo, a manera de que sea más corto el código y por lo tanto optimicemos el trabajo.

La global es la más lenta, ya que se trata de ver todo el código.

La de mirilla es parecida a la de los ciclos, pero esta se enfoca en las decisiones y saltos de rutinas. Recorre el código buscando combinaciones que pueden ser reemplazadas por otras mucho más rápidas.

Por los costos siempre hay que ver la mejor forma de optimizar el trabajo ya que simplemente por la pura optimización puede llegar a ser caro. El costo de ejecución es lo que implica en donde se va a correr el programa, según yo viene siendo el hardware que se ocupa para que se ejecute bien. Por ejemplo, los videojuegos.

Para la optimización del código hay que decirles siempre a los programadores desde el inicio que traten de que su código sea optimo, y que traten de colocar líneas de más.