Instituto Tecnológico Superior de Jerez

Ingeniería en Sistemas Computacionales

Lenguajes y Autómatas 2

Semestre: 7°

Actividad: Ensayo de optimización

Tema 3

Docente: ISC. Jesús Aranda Gamboa

Alumno: Marco Enrique Villegas Ulloa
No. Control: 14070094

23/10/2019

Técnicas para la Optimización del Código

Introducción

En el momento en el cual un compilador de código analiza el código genera un código intermedio. Este mismo código al momento de estar generándolo aplica diferentes métodos para optimizar el código y hacerlo mas legible para la máquina.

Estos métodos utilizados por el compilador para aligerar el trabajo para la maquina para leerlo son variados y cada compilador utiliza diferentes estrategias para ello, pero en general utilizan las mismas bases para optimizar el código.

Optimización de código

En la optimización de un código se optimiza los recursos según lo que se busca, si se busca optimizar tiempo u optimizar memoria, también se puede optimizar ambos recursos. En la fase de optimización de busca que sea rápido, entonces, se busca un punto medio de mejora al código ya que puede tomar demasiado tiempo optimizando estos recursos del código convirtiéndolo en un proceso lento en nuestro compilador perdiendo en si eficiencia. También al momento de mejorar el código es muy importante no modificar el sentido del código, en otras palabras, al momento que una línea es cambiada debe tener la misma esencia, entrada y salida de datos si no corrompería el código y su propósito.

Un método es **Optimización independiente de la maquina** el cual implica en optimizar lo más posible los registros del CPU innecesarios como por ejemplo si declaras una variable dentro de un ciclo esta variable se declarará cada vez que el ciclo haga una iteración esto consume memoria en el sistema. Tomando esto en cuenta la mejor forma de ahorrar recursos es relocalizando la declaración de la variable afuera del ciclo de esta manera solo se declarará una vez y se buscara en afectar lo menos posible al ciclo si es necesario se remplaza la antigua declaración a una igualación de valor para que en cada iteración cumpla su objetivo, de esta manera ahorramos registros en el CPU optimizando el código.

En la **Optimización dependiente de la maquina** se realiza después de generar el código destino y cuando el código se genera de acuerdo a la arquitectura de la máquina de destino se busca aprovechar al máximo la jerarquía de la memoria del equipo. Modificando registros para dar un orden a la jerarquía.

Se le considera un **Bloque básico** a una sección de código que se ejecuta siempre en orden y no tiene saltos. Al momento de ejecutar la instrucción todas las instrucciones del mimo bloque será ejecutado en secuencia en aparición sin perder el control de flujo del programa. Un bloque básico puede ser condicionales como *IF-ELSE* también bucles como *FOR, DO-WHILE*.

Podemos utilizar el siguiente algoritmo para encontrar los bloques básicos en un programa:

- *Declaraciones del cabezal Búsqueda de todos los bloques básicos desde donde se inicia un bloque básico:*
    - *Primera declaración de un programa.*
    - *Las declaraciones que son objetivo de cualquier rama (condicional o incondicional).*
    - *Las declaraciones que siguen cualquier rama.*
- *Las declaraciones del cabezal y las declaraciones siguientes forman un bloque básico.*

- *Un bloque básico no incluye cualquier cabezal declaración de cualquier otro bloque básico. (tutorialspoint)*

En una **Optimización de bucle** se busca disminuir lo mas posible las iteraciones del ciclo en el código para ello se utilizan diferentes técnicas.

si un cacho del código en el bucle es una línea que no varía se puede localizar afuera del bucle para ahorrar memoria esto se le conoce como *Código Invariante.*

Si una variable es modificada su valor dentro del bule del bucle entonces se conoce como *inducción variable.*

En los ciclos es preferente cuando es posible remplazarlos por las expresiones mas simples que cuesten menos ciclos sin comprometer la estructura del código como por ejemplo *(x * 2)* esto cuesta recursos en ciclos, pero una expresión equivalente y menos costosa seria *(x<<1)* esta operación binaria casi no consume recursos. Esto se le conoce como *Fuerza reducción*.

Una forma de optimización también es **Eliminar código muerto** se les conoce código muerto a fragmentos de código que nunca se ejecutan, son inalcanzables o no tiene valor alguno en el código. Estas líneas pueden ser eliminadas siempre y cuando no afecte a la estructura del código.

Código muerto parcialmente

Es cuando en una condicional nuca se va a dar una de sus posibilidades, o cuando la asignación de una variable esta dentro de una condicional, pero resulta tan insignificante la asignación de valor posible que no afecta en absoluto el programa estos son candidatos para ser eliminados del código.

Conclusión

En el momento que la compilación esta en marcha es una fase importante la optimización para que ahorre recursos de la máquina, de esta forma no se saturaría la memoria o tarde innecesariamente mas de lo divido por tontos errores de programación básicos.

Para mi consideración si es importante que esta fase sea rápida para que no alenté al compilador. Pero si se quiere hacer una mayor optimización seria utilizando prácticamente una inteligencia artificial simple para que evalué por completo el código, sino muchas reglas las cuales debe seguir para una buena optimización.

Referencias

Informática. (2019). *Optimización de código: un código más eficiente - Informática*. [online] Available at: http://informatica.blogs.uoc.edu/2016/05/02/optimizacion-de-codigo-un-codigo-mas-eficiente/ [Accessed 24 Oct. 2019].

Tutorialspoint.com. (2019). *Optimizacion de codigo*. [online] Available at: https://www.tutorialspoint.com/es/compiler_design/compiler_design_code_optimization.htm [Accessed 24 Oct. 2019].

Sites.google.com. (2019). *2.5. Generación de código intermedio - Teoría de Lenguajes y Compiladores*. [online] Available at: https://sites.google.com/site/teoriadelenguajesycompiladores/procesadores-de-lenguaje/generacion-de-codigo [Accessed 24 Oct. 2019].