SSC0951 **Desenvolvimento de Código Otimizado**

Atividade 4 **Otimização do Compilador**

Nome: **Gabriel Van Loon Bodê da Costa Dourado Fuentes Rojas**

Alberto Campos Neves

---

1. Introdução

Nesta prática iremos analisar o efeito de diferentes opções de otimizações que os compiladores modernos oferecem à seus usuários na hora de transformar o código fonte em código de máquina. Para isso estaremos usando o programa *`fannkuch-redux`* escolhido dentre as opções oferecidas no website "Computer Language Benchmarks Game" utilizando a linguagem C++ com as seguintes configurações:

- **Compilador:** g++ (Ubuntu 9.3.0-10ubuntu2) 9.3.0
- **Arquitetura:** intel x86_64 (Intel(R) Core(TM) i5 CPU 650 @ 3.20GHz)
- **Sistema Operacional:** Ubuntu
- **Versão do Kernel:** Linux 5.4.0-33-generic

2. Planejamento de Experimento

Devido à grande quantidade de flags que decidimos testar, percebemos que seria inviável utilizar um planejamento fatorial para explorar a influência das flags e decidimos utilizar uma abordagem de planejamento simples, sendo o nosso base os resultados do programa compilado sem o uso de nenhuma flag.

As flags sendo utilizadas como fatores são as seguintes: ***-O1, -O2, -O3,*** *-finline, -fstrict-aliasing, -ftree-pre, -ftree-vrp, -ftree-ch, -freorder-functions, -fcaller-saves, -freorder-blocks, -finline-small-functions, -fomit-frame-pointer, -fguess-branch-probality, -fpartial-inlining, -foptimize-sibling-calls, -fdce, -fpeephole2, -fif-conversion, -ftree-partial-pre, -freorder-function.* Os detalhes de cada uma destas flags podem ser devidamente consultados no manual do compilador utilizado.

Para cada fator, foi calculado o tempo de execução em segundos com uma precisão de até 6 casas decimais. Além disso, fizemos também o cálculo da variação da de cada fator

com relação ao caso base utilizando a mesma fórmula vista no artigo utilizado como referência.

$$Variação(\%) = \frac{Tempo_{caso\ base} - Tempo_{fator}}{Tempo_{caso\ base}}$$

Veja que é possível a presença de variações negativas, uma vez que é totalmente plausível existir otimizações que afetem a performance negativamente.

3. Resultados Obtidos

Na tabela abaixo exibimos os resultados obtidos com o caso base e em seguida com os diferentes fatores ordenados por ordem decrescente de variação.

Flag	Tempo de Execução (*seg*)	Variação (%)
Caso Base (nenhuma flag)	**14.077767**	**0.000000**
-O3	**2.127529**	**0.848873**
-O2	**2.581083**	**0.816655**
-O1	**2.895254**	**0.794339**
Todas as flags (*)	**12.15593**	**0.136515**
-fomit-frame-pointer	**11.936843**	**0.152078**
-ftree-vrp	**13.987449**	**0.006416**
-ftree-ch	**13.993222**	**0.006006**
-fguess-branch-probability	**14.001763**	**0.005399**
-foptimize-sibling-calls	**14.003107**	**0.005303**
-fstrict-aliasing	**14.005320**	**0.005146**
-fcaller-saves	**14.009683**	**0.004836**
-fdce	**14.014277**	**0.004510**
-freorder-blocks	**14.021394**	**0.004004**
-ftree-partial-pre	**14.027943**	**0.003539**
-fpartial-inlining	**14.036853**	**0.002906**
-fpeephole2	**14.037513**	**0.002859**

-ftree-pre	**14.042101**	**0.002533**
-finline	**14.065316**	**0.000884**
-freorder-functions	**14.140258**	**-0.004439**
-finline-small-functions	**14.111786**	**-0.002416**
-fif-conversion	**14.110979**	**-0.002359**

Tabela 1: Resultados obtido do planejamento ordenamos pela ordem de variação decrescente.

4. Conclusões

O primeiro resultado que podemos concluir que já era esperado se trata da ordem em que os fatores -O3, -O2 e -O1 se encontram, uma vez que quanto maior o valor mais agressiva são as mudanças efetuadas pelo compilador e maior é a otimização realizada pelo mesmo.

Quanto às flags utilizadas individualmente, vemos que a ***-ftree-ch*** é uma das responsáveis pela maior variação em relação ao código base, e isso também ocorre no nos resultados da referência [1]. A flag ***-finline***, no entanto, aparenta fazer uma variação negativa, o que contrasta com os resultados esperados baseado na nossa referência. Não conseguimos chegar num consenso do motivo isto ocorre.

Uma outra flag que foi incluída e é a que possui a maior variação, é a ***-fomit-frame-pointer***, no entanto ela não parece se encontrar dentro das opções -OX e decidimos avaliá-la porque no site em que retiramos o programa de benchmark ela estava sendo acionada em conjunto com a -O3. Decidimos, portanto, averiguar sua variação em relação às outras que já haviam sido escolhidas.

(*) Um outro teste que também decidimos fazer posteriormente foi o de compilar o programa com todas as flags com 80% (mesmo aquelas que indicaram variação negativa) para verificar qual seria a influência de todas sendo aplicadas simultaneamente. Como podemos ver, o resultado de todas combinado ainda é inferior à quando apenas o ***-fomit-frame-pointer*** é aplicado, não conseguimos pensar em uma justificativa para este fenômeno.

Por fim, conseguimos ver que grande parte das flags com variação positiva obteve uma melhoria similar em relação ao tempo base, ficando claro a importância da combinação

destas flags em um planejamento fatorial para investigar também os efeitos quando as mesmas estão sendo correlacionadas.

5. Referências

[1] Luque G., Alba E. (2018) Finding Best Compiler Options for Critical Software Using Parallel Algorithms. In: Del Ser J., Osaba E., Bilbao M., Sanchez-Medina J., Vecchio M., Yang XS. (eds) Intelligent Distributed Computing XII. IDC 2018. Studies in Computational Intelligence, vol 798. Springer, Cham.