Os efeitos do paralelismo ao nível de instrução (ILP)

Análise Experimental com Otimizações de Compilador

Werbert Arles de Souza Barradas

Universidade Federal do Rio Grande do Norte (UFRN)
Disciplina de Programação Paralela - DCA3703

August 24, 2025

Introdução

Contexto

As arquiteturas de processadores modernos dependem do **Paralelismo em Nível de Instrução (ILP)** para alcançar alto desempenho, executando múltiplas instruções por ciclo de clock.

O Problema: Dependência de Dados

A capacidade de explorar o ILP é severamente limitada por dependências de dados no código, que forçam uma execução sequencial e subutilizam o hardware.

Objetivo do Estudo

Investigar e quantificar o impacto destas dependências e analisar como a estrutura do código e as otimizações do compilador interagem para explorar o ILP.

Metodologia do Experimento

- **Programa:** Desenvolvido em C, opera sobre um vetor de 100 milhões de inteiros.
- **Laços Analisados:** Quatro laços foram testados para exibir diferentes características de dependência:
  - **Laço 1:** Inicialização de vetor (iterações independentes).
  - **Laço 2:** Soma com dependência de dados (acumulador único).
  - **Laço 3:** Soma com quebra de dependência (Loop Unrolling, fator 4).
  - **Laço 4:** Tentativa de maximizar o ILP (Loop Unrolling, fator 8).
- **Compilação:** O código foi compilado com GCC sob três níveis de otimização: `-O0`, `-O2` e `-O3`.
- **Medição:** O tempo de execução de cada laço foi medido com alta precisão usando `clock_gettime`.

Analisando o Código: Dependência vs. Independência

Laço 2: O Gargalo (Dependência)

Uma longa cadeia de dependência na variável `soma_dependente` força a execução sequencial. O pipeline da CPU precisa parar (*stall*) a cada iteração.

```
long long soma_dependente = 0;
for (int i=0; i<TAMANHO_VETOR; i++) {
    soma_dependente += vetor[i];
}
```

Analisando o Código: Dependência vs. Independência

Laço 3: A Solução (Independência)

O uso de 4 acumuladores quebra a cadeia. As 4 somas dentro do laço são independentes e podem ser executadas em paralelo pela CPU, explorando o ILP.

```
long long s1=0, s2=0, s3=0, s4=0;
for (int i=0; i<TAMANHO_VETOR; i+=4) {
    s1 += vetor[i];
    s2 += vetor[i+1];
    s3 += vetor[i+2];
    s4 += vetor[i+3];
}
long long soma_total = s1+s2+s3+s4;
```

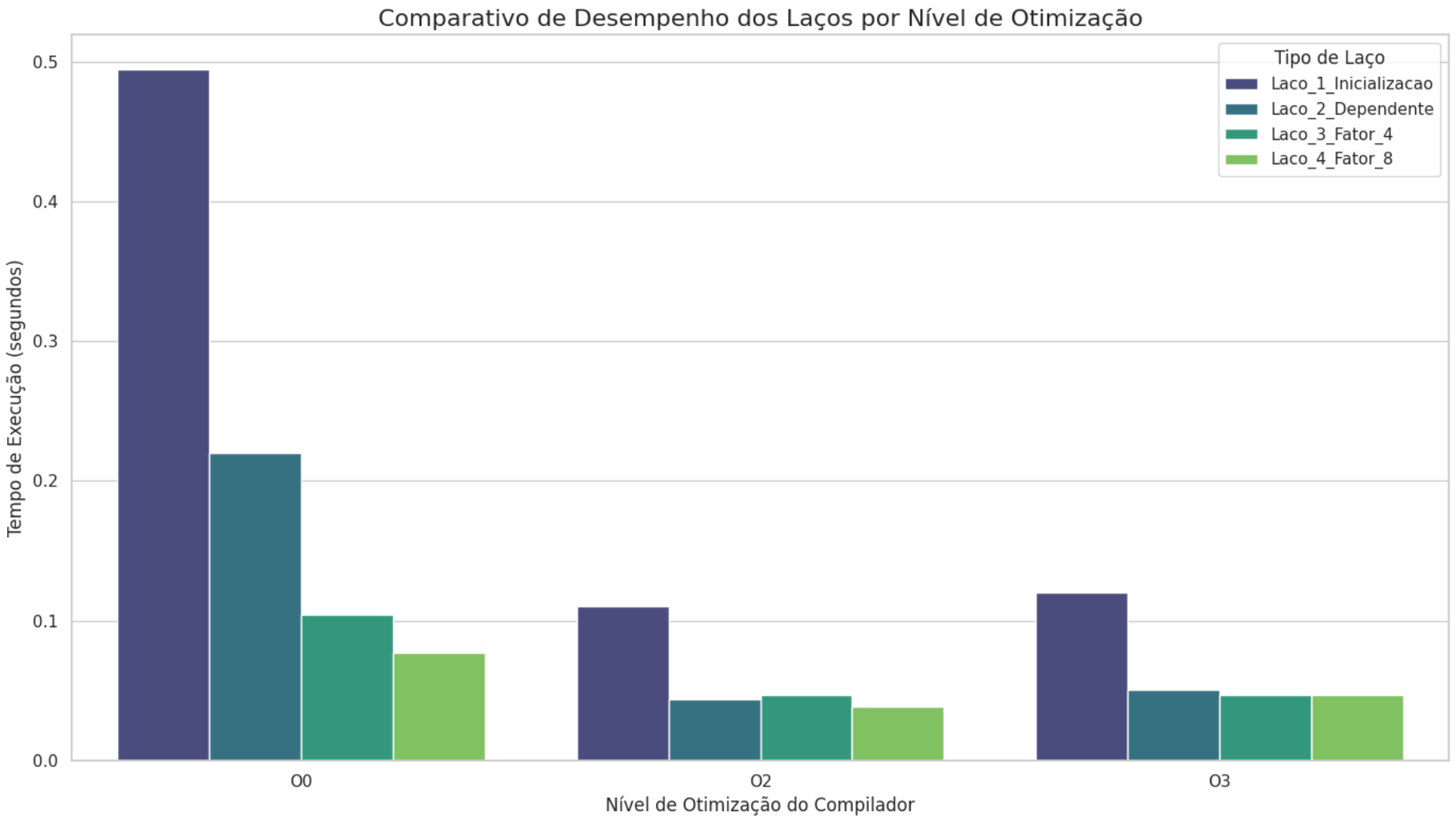
Analisando o Código: Dependência vs. Independência

Laço 4: A Solução (Independência)

O uso de 8 acumuladores quebra a cadeia. As 8 somas dentro do laço são independentes e podem ser executadas em paralelo pela CPU, explorando o ILP.

```
long long s1=0, s2=0, s3=0, s4=0, s5=0, s5=0, s70, s8=0;
for (int i=0; i<TAMANHO_VETOR; i+=8) {
    a1 += vetor[i];
    a2 += vetor[i+1];
    a3 += vetor[i+2];
    a4 += vetor[i+3];
    a5 += vetor[i+4];
    a6 += vetor[i+5];
    a7 += vetor[i+6];
    a8 += vetor[i+7];
}
long long soma_total = s1+s2+s3+s4;
```

Tempos de execução (em segundos) por laço e nível de otimização



Como o Desempenho foi Calculado?

Definição

A performance foi medida em **GFLOPS**: **Giga** (bilhões de) **F**loating-point **O**perations **P**er **S**econd. Embora a operação seja com inteiros, esta é uma métrica padrão para avaliar a capacidade de processamento computacional.

Fórmula Geral

A fórmula para calcular o desempenho é:

$$\text{GFLOPS} = \frac{\text{Número Total de Operações}}{\text{Tempo de Execução (s)} \times 10^9}$$

Aplicação no Projeto

- **Tamanho do Vetor (N):** 100.000.000.
- **Operações por Iteração:** A operação `soma += vetor[i]` consiste em 2 operações: 1 leitura da memória e 1 adição.
- **Total de Operações:** $2 \times N = 200.000.000$.

Desempenho (GFLOPS) vs. Nível de Otimização (com Média)

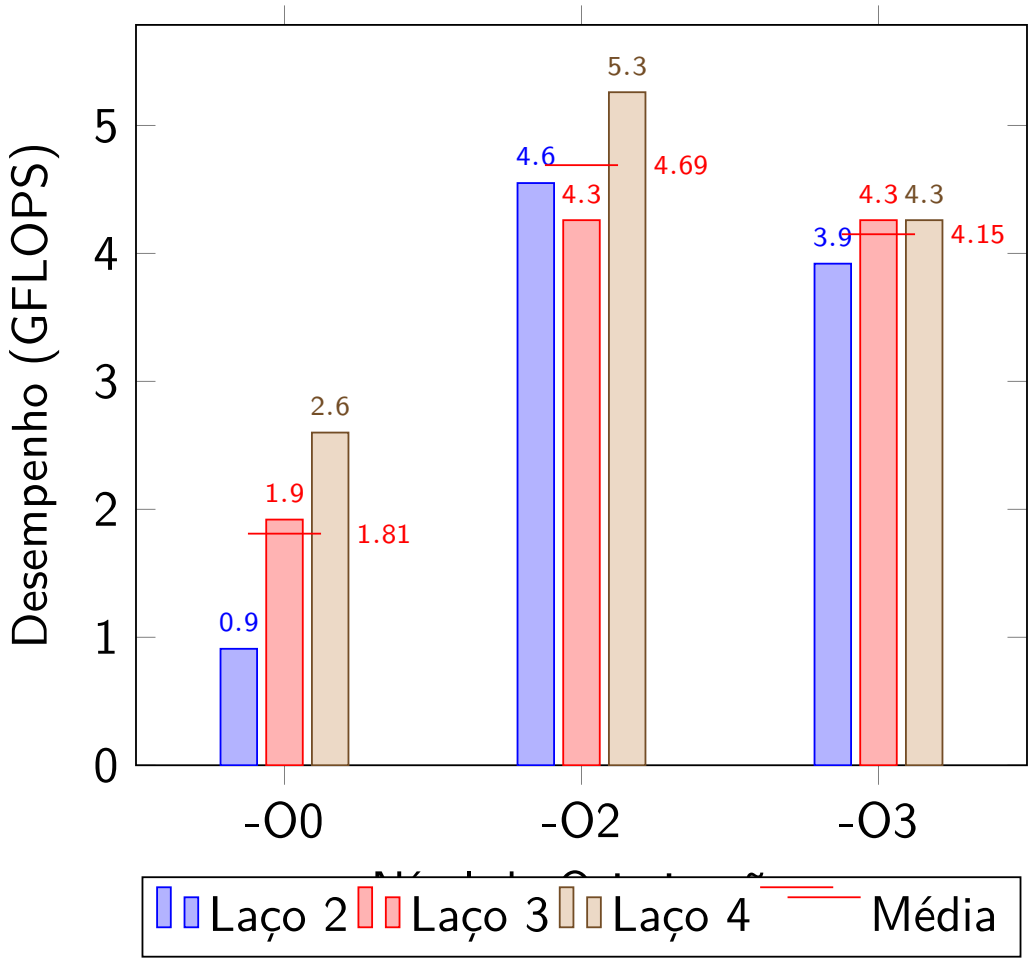


Figure: As marcas vermelhas indicam o desempenho médio.

Interpretação dos Resultados (GFLOPS)

Salto de -O0 para -O2: O Poder da Otimização

Como visto no gráfico anterior, a ativação das otimizações do compilador resulta num ganho de performance massivo, de 4 a 5 vezes. O desempenho médio salta de ~1 GFLOP para mais de 4.5 GFLOPS.

Pico de Desempenho em -O2: Sinergia Código-Compilador

O maior desempenho de todo o experimento (**5.3 GFLOPS**) é alcançado pelo "Laço 4" (unroll x8) compilado com -O2. Isso sugere que a estrutura de código que expõe o ILP manualmente, combinada com as otimizações moderadas, criou o cenário ideal para o hardware.

Interpretação dos Resultados (GFLOPS)

Convergência em -O3: O Limite do Hardware

Com a otimização agressiva (-O3), o desempenho de todos os laços converge para ~4.15 GFLOPS. O compilador torna-se tão eficiente que consegue otimizar até mesmo o código com dependência de dados, mas o desempenho geral não supera o pico de -O2, indicando que o gargalo passa a ser o próprio hardware.

Conclusão

Interpretação dos Resultados

A análise dos dados leva às seguintes conclusões:

- **Estrutura do Código é Crucial:** Sem otimizações (-O0), a estrutura do código que quebra dependências (Laços 3 e 4) é fundamental para permitir que o hardware explore o ILP, resultando em ganhos de performance de mais de 280%.
- **O Poder do Compilador:** Com otimizações (-O2), o compilador consegue reordenar e otimizar o código do Laço 2, diminuindo a penalidade da dependência de dados. Ainda assim, o Laço 4 (unroll x8) se mantém como o mais rápido.
- **Otimização Agressiva e Limites do Hardware:** Em -O3, o compilador aplica técnicas tão agressivas (como auto-vetorização) que o desempenho do laço simples se iguala ao das versões com *unrolling* manual. Isso indica que o gargalo deixa de ser o código e passa a ser os limites físicos do próprio processador.