Programação de Alto Desempenho

Atividade 4 - Otimizações independentes de compilador ou maquina

Lucas Santana Lellis - 69618
PPGCC - Instituto de Ciência e Tecnologia
Universidade Federal de São Paulo

I. Introdução

Nesta atividade foram realizados experimentos relacionados com a otimização de código independente de compilador ou máquina, e também na utilização de recursos básicos do gprof. Cada experimento foi realizado 5 vezes, e os resultados apresentados são a média dos resultados obtidos em cada um deles.

Todos os programas foram feitos em C, com otimização -O3, utilizando a biblioteca PAPI para estimar o tempo total de processamento, e o número de operações por ciclo.

As especificações da máquina utilizada estão disponíveis na Tabela I.

Tabela I: Especificações da Máquina

CPU	Intel Core i5 - 3470
Cores	4
Threads	4
Clock	3.2 GHz
Cache L3	6144 KB
Cache L2	256 KB * 4
Hardware Counters	11
RAM	8 Gb
SO	Fedora 23
Kernel	4.7.4
GCC	5.3.1

II. Experimento I - Otimização do código do Jogo da Vida

Neste experimento foi realizada a comparação da alteração de desempenho causada pela redução do número de multiplicações no endereçamento dos elementos da matriz utilizada para representar o jogo da vida.

Assim, no primeiro programa, chamado de "Padrão", o cálculo de cada vizinho de uma coordenada (i, j) envolve multiplicações e somas, como o código abaixo:

```
up = val[(i-1)*n + j];
upright = val[(i-1)*n + j+1];
right = val[i*n + j+1];
rightdown = val[(i+1)*n + j+1];
down = val[(i+1)*n + j];
downleft = val[(i+1)*n + j-1];
left = val[i*n + j-1];
leftup = val[(i-1)*n + j-1];
```

Enquanto o segundo programa, chamado de "Otimizado", o cálculo da posição atual é dado por $inj = i * n + j$, e os vizinhos são obtidos pelo deslocamento em função da soma e subtração de valores conhecidos:

```
int inj = i*n + j;
up = val[inj - n];
upright = val[inj - n + 1];
right = val[inj + 1];
rightdown = val[inj + n + 1];
down = val[inj + n];
downleft = val[inj + n - 1];
left = val[inj - 1];
leftup = val[inj - n - 1];
```

Tendo como configuração inicial um tabuleiro de tamanho 10x10 com células vivas nas posições (1,2), (2,3), (3,1), (3,2), (3,3) e as demais células mortas. Nas Figuras de 1 até 5 a configuração do tabuleiro nas quatro primeiras iterações do algoritmo.

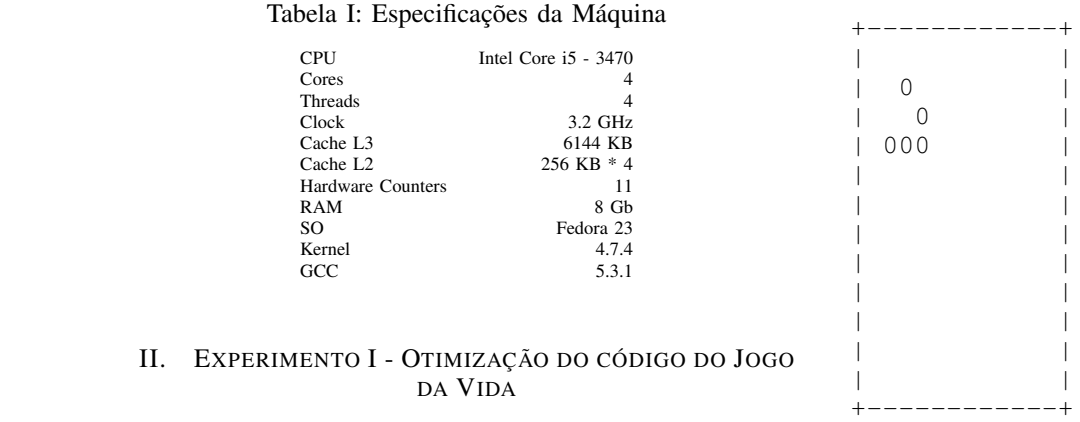

Figura 1: Condições iniciais do teste em tabuleiro 10x10.

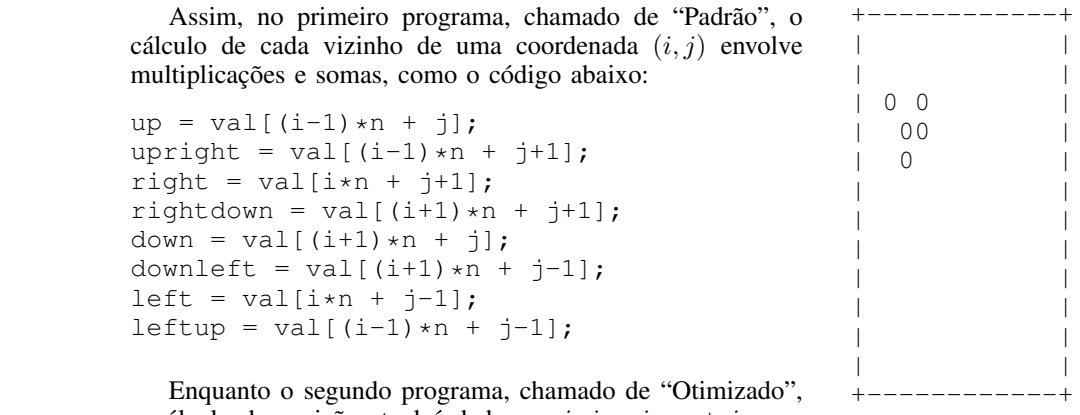

Figura 2: Condições do tabuleiro 10x10 após 1a iteração.

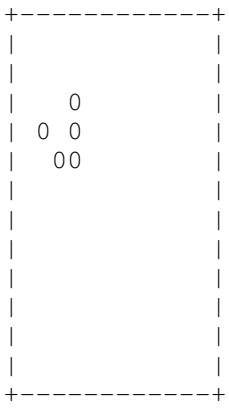

Figura 3: Condições do tabuleiro 10x10 após 2a iteração.

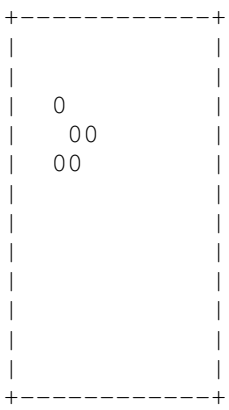

Figura 4: Condições do tabuleiro 10x10 após 3a iteração.

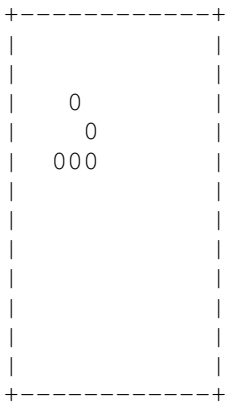

Figura 5: Condições do tabuleiro 10x10 após 4a iteração.

Foram realizados também testes em tabuleiros de tamanho 1000x1000, de forma que na Figura 6 temos as 10 primeiras linhas e colunas da configuração inicial do tabuleiro e na Figura 5, as 10 últimas linhas e colunas da geração final do tabuleiro após $4(N-5)$ iterações do algoritmo.

Finalmente, na Tabela II temos a avaliação do desempenho do algoritmo Original e Modificado, após $4(N-3)$ iterações, e percebemos o tempo de execução foi ligeiramente maior,

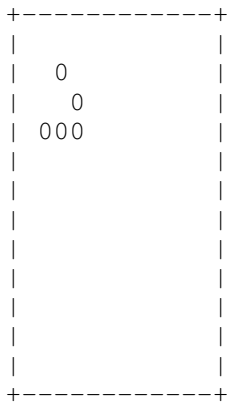

Figura 6: Condições iniciais do tabuleiro 10x10.

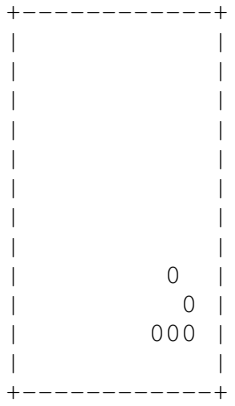

Figura 7: Condições do tabuleiro 1000x1000 após 4(N-5) iterações.

assim como o número de ciclos por elemento, porém, percebe-se que esse tipo de otimização não surte grande efeito, já que a variação foi menor do que um décimo de segundo.

Mode	**Tempo(s)**	**CPE**
Padrao	7.5182	26185.8800
Otimizado	7.5660	26377.8400

Tabela II: Avaliação do desempenho do algoritmo Original e Modificado.

III. Experimento II - GPROF

O segundo experimento consiste na simples utilização do gprof para verificar os tempos de execução por rotina do algoritmo original. Como só há uma função, a função "evolve" tem 100% do tempo de execução, como visto na Figura 8:

```
Flat profile:

Each sample counts as 0.01 seconds.
%   cumulative   self              self     total
time   seconds   seconds    calls  Ts/call  Ts/call  name
100.54      7.52     7.52                             evolve
```

Figura 8: Saída do profiler gprof, após a execução do algoritmo original do jogo da vida.