Atividade 1 de desenvolvimento de código otimizado

Allan Garcia Cavalcante e Silva, 13731222
Eduarda Fritzen Neumann, 12556973
Lucas Eduardo Gulka Pulcinelli, 12547336
Teo Sobrino Alves, 12557192

Setembro de 2023

1 Descrição da atividade

A implementação da multiplicação de matrizes quadradas de dimensão 500 foi implementada de 6 maneiras distintas, utilzando alocação estática ou dinâmica e otimizações de *loop interchange*, *loop unrolling* ou sem otimizações.

```c
int main()
{
  int a[n * n];
  int b[n * n];
  int c[n * n];
  multiplyNo(c, a, b);
}
```

Implementação estática

```c
int main()
{
    int *a = malloc(sizeof(int) * n * n);
    int *b = malloc(sizeof(int) * n * n);
    int *c = malloc(sizeof(int) * n * n);
    multiplyNo(c, a, b);
    free(a);
    free(b);
    free(c);
}
```

Implementação dinâmica

```c
void multiplyNo(int *c, const int *a,
                const int *b)
{
  for (int i = 0; i < n; i++){
    for (int j = 0; j < n; j++){
      for (int k = 0; k < n; k++){
        c[i*n+j]=a[i*n+k]*b[k*n+j];
      }
    }
  }
}
```

Multiplicação não otimizada

```c
void multiplyLi(int *c, const int *a,
                const int *b)
{
  for (int i = 0; i < n; i++){
    for (int k = 0; k < n; k++){
      for (int j = 0; j < n; j++){
        c[i*n+j]=a[i*n+k]*b[k*n+j];
      }
    }
  }
}
```

loop interchange

```c
void multiplyLu(int *c, const int *a, const int *b)
{
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            int k;
            for (k = 0; k < n / 10;){
                c[i * n + j] = a[i * n + k] * b[k * n + j];
                k++;
                c[i * n + j] = a[i * n + k] * b[k * n + j];
                k++;
                c[i * n + j] = a[i * n + k] * b[k * n + j];
                k++;
                c[i * n + j] = a[i * n + k] * b[k * n + j];
                k++;
                c[i * n + j] = a[i * n + k] * b[k * n + j];
                k++;
                c[i * n + j] = a[i * n + k] * b[k * n + j];
                k++;
                c[i * n + j] = a[i * n + k] * b[k * n + j];
                k++;
                c[i * n + j] = a[i * n + k] * b[k * n + j];
                k++;
                c[i * n + j] = a[i * n + k] * b[k * n + j];
                k++;
                c[i * n + j] = a[i * n + k] * b[k * n + j];
                k++;
            }
            for (; k < n; k++){
                c[i * n + j] = a[i * n + k] * b[k * n + j];
            }
        }
    }
}
```

loop unrolling

Cada experimento realizado é detalhado na tabela abaixo.

Experimento	Alocação	Otimização	Arquivo implementado
1	Estática	None	`static-no.c`
2	Dinâmica	None	`dynamic-no.c`
3	Estática	*loop interchange*	`static-li.c`
4	Dinâmica	*loop interchange*	`dynamic-li.c`
5	Estática	*loop unrolling*	`static-lu.c`
6	Dinâmica	*loop unrolling*	`dynamic-lu.c`

Os testes foram realizados com o comando `perf stat` utilizando a flag `-r` para que cada teste fosse executado diversas vezes, `-e` para coletar informações sobre os eventos desejados e `-o` para armazenar os resultados no arquivo informado. Dessa forma, o comando abaixo foi usada para executar o teste com alocação estática e *loop interchange*, por exemplo.

```
perf stat -r 10  -o ../perf-out/static-li.txt -e L1-dcache-loads,
L1-dcache-load-misses,branch-instructions,branch-misses
../bin/static-li
```

O script `make_csv.py` foi usado para processar as saídas geradas e compilar as informações dos testes no arquivo `experiment_results.csv`.

Os intervalos de confiança do tempo foram calculados com o script `calculate_confidence_intervals.py`, utilizando a bilbioteca `scipy.stats`. Assumiu-se que o tempo segue a distribuição *t-sudent* e utilizou-se um p-valor de 5%. O erro padrão da média (SEM) de cada experimento foi calculado com base no desvio padrão (σ) informado pelo comando `perf stat` e pela quantidade de medidas tomadas (n), nesse caso 10, por meio da fórmula $SEM = \sigma/\sqrt{n}$. Os resultados foram registrados no arquivo `experiment_results.csv`.

```
static_no_sem = static_no_stddev / np.sqrt(len)
st.t.interval(confidence=0.95, df=len-1, loc=static_no_mean,
              scale=static_no_sem)
```

A influência dos fatores foram calculadas com o script `calculate_factors_influence.r`, desenvolvido com base no script disponíbilizado, e o resultado foi salvo no arquivo `factors_influence.csv`. Foram realizadas 4 análises, todas do tipo 2 fatorial completo, descritas na tabela abaixo.

Análise	**Nível alocação**		**Nível otimização**		**Variável resposta**
1	Estática	Dinâmica	None	*interchange*	L1-dcache-loads
2	Estática	Dinâmica	None	*interchange*	L1-dcache-loads-misses
3	Estática	Dinâmica	None	*unrolling*	branch-instructions
4	Estática	Dinâmica	None	*unrolling*	branch-misses

2 Resultados dos experimentos

Para cada experimento, ao final das 10 repetições obteve-se a média dos valores dos eventos observados, além da média e desvio padrão do tempo de execução. Os valores são apresentados abaixo.

2.1 Experimento 1

Evento observado	**Média obtida**
L1-dcache-loads	2126350811
L1-dcache-loads-misses	16808
branch-instructions	125789160
branch-misses	253262

Tempo médio de 0.45796s com desvio padrão de 0.00218s. Portanto, o intervalo de confiança com 95% de certeza para o tempo é (0.45567, 0.46025), em segundos.

2.2 Experimento 2

Evento observado	**Média obtida**
L1-dcache-loads	2126382471
L1-dcache-loads-misses	37255
branch-instructions	125790164
branch-misses	253745

Tempo médio de 0.45723s com desvio padrão de 0.00171s. Portanto, o intervalo de confiança com 95% de certeza para o tempo é (0.45544, 0.45902), em segundos.

2.3 Experimento 3

Evento observado	**Média obtida**
L1-dcache-loads	2126352637
L1-dcache-loads-misses	22523
branch-instructions	125789112
branch-misses	253258

Tempo médio de 0.41892s com desvio padrão de 0.00225s. Portanto, o intervalo de confiança com 95% de certeza para o tempo é (0.41656, 0.42128), em segundos.

2.4 Experimento 4

Evento observado	Média obtida
L1-dcache-loads	2126364691
L1-dcache-loads-misses	44624
branch-instructions	125789904
branch-misses	253571

Tempo médio de 0.41976s com desvio padrão de 0.00212s. Portanto, o intervalo de confiança com 95% de certeza para o tempo é (0.41754, 0.42198), em segundos.

2.5 Experimento 5

Evento observado	Média obtida
L1-dcache-loads	2104350990
L1-dcache-loads-misses	16947
branch-instructions	115039169
branch-misses	289709

Tempo médio de 0.45838s com desvio padrão de 0.00204s. Portanto, o intervalo de confiança com 95% de certeza para o tempo é (0.45624, 0.46052), em segundos.

2.6 Experimento 6

Evento observado	Média obtida
L1-dcache-loads	2104358683
L1-dcache-loads-misses	33950
branch-instructions	115039948
branch-misses	294994

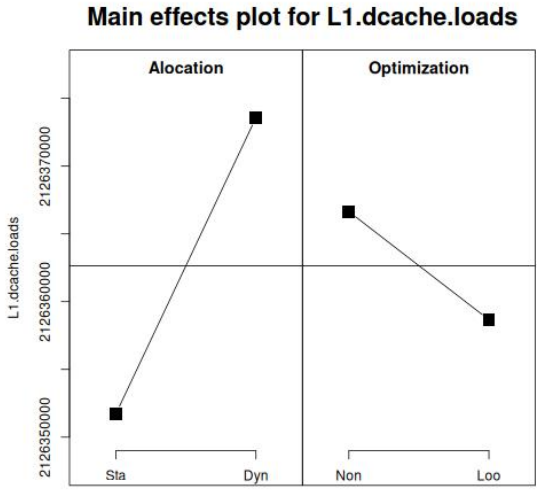
Tempo médio de 0.454109s com desvio padrão de 0.000386s. Portanto, o intervalo de confiança com 95% de certeza para o tempo é (0.4537, 0.45451), em segundos.

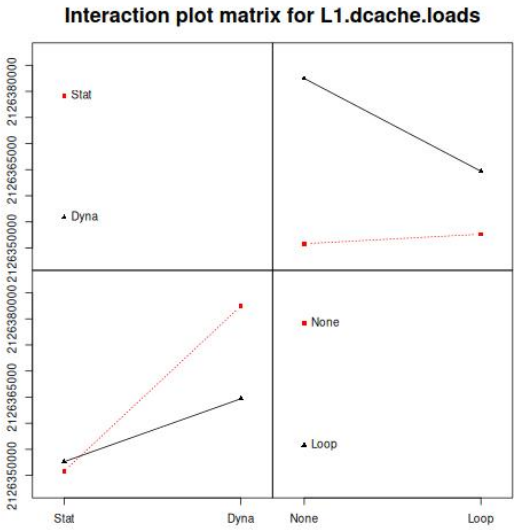
3 Influência dos fatores

Para cada análise foi calculada a influência individual de cada fator e a influência conjunta. Também, foram gerados os gráficos de efeito principal, onde pode-se comparar os níveis de um mesmo fator, desconsiderando o outro fator, e de interação, onde, dado um nível de um fator, pode-se comparar sua interação com os níveis do outro fator.

3.1 Análise 1

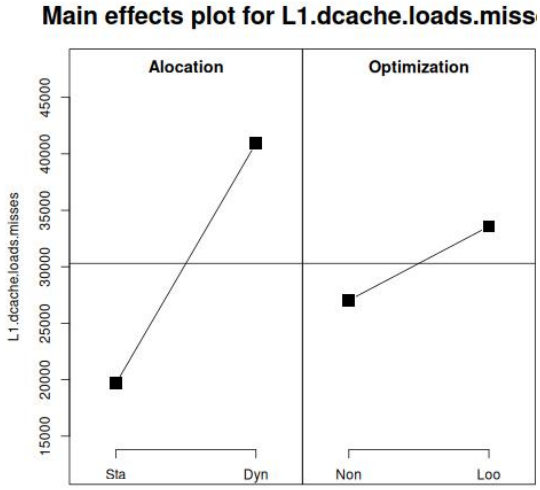
Fator	Influência
Alocação	0.749425232372584
Otimização	0.0998220284573909
Combinada	0.150752739170025

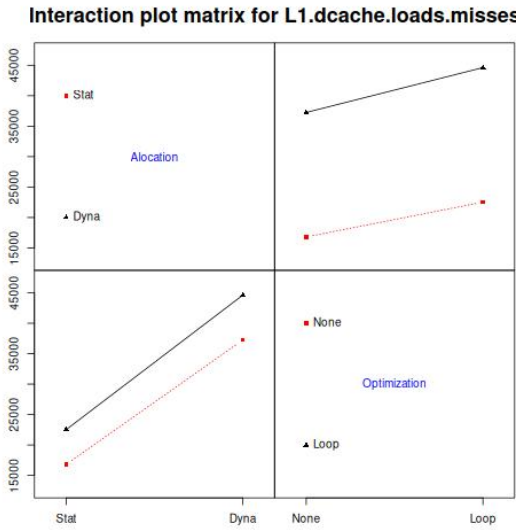




3.2 Análise 2

Fator	Influência
Alocação	0.91234674236662
Otimização	0.086274548555977
Combinada	0.00137870907740274

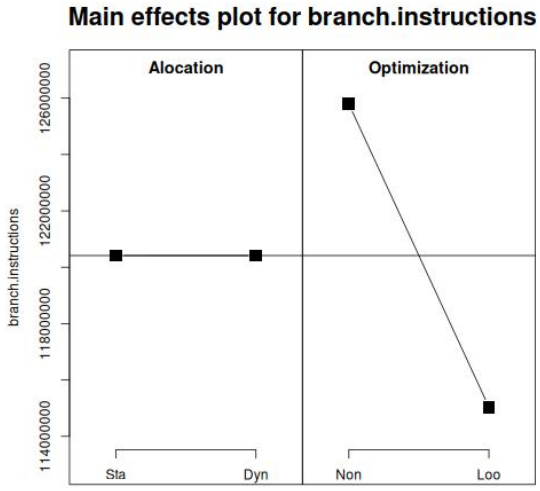


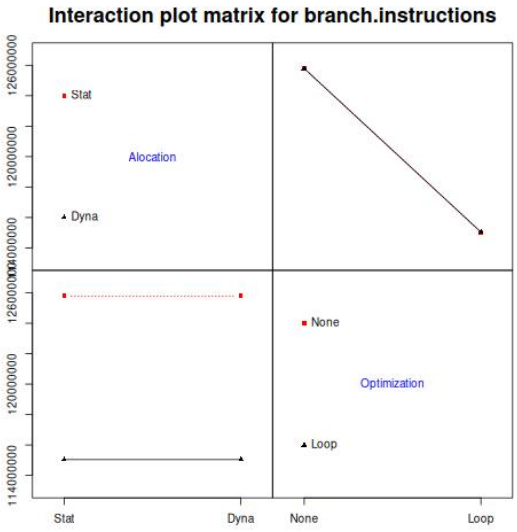


3.3 Análise 3

A influência na variável *branch-instructions* da alocação foi 6.87729099568712e-09, da otimização foi 0.999999993013192 e combinada foi 1.09516549136072e-10.

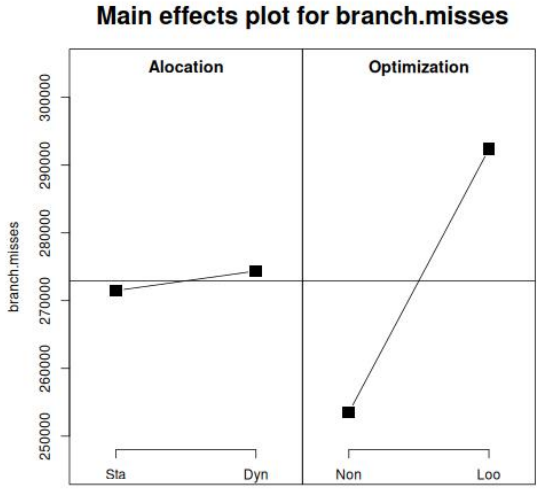
Fator	Influência
Alocação	6.87729099568712e-09
Otimização	0.999999993013192
Combinada	1.09516549136072e-10

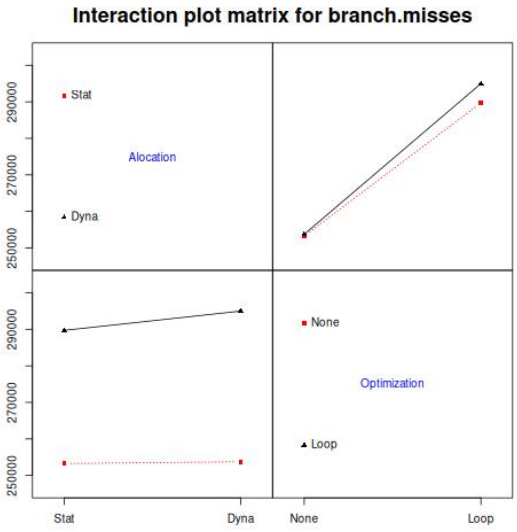




3.4 Análise 4

Fator	Influência
Alocação	0.00546033775752893
Otimização	0.990755120362726
Combinada	0.00378454187974538




Interaction plot matrix for branch.misses
Stat
Alocation
Dyna
None
Optimization
Loop
290000
270000
250000
Stat
Dyna
None
Loop

4 Discussão dos Resultados

4.1 Comparação do tempo

Observa-se que o tempo de execução é consideravelmente menor nos experimentos 3 e 4, com otimização *loop interchange*. Fixando os níveis do fator otimização, não há um impacto perceptível causado no tempo pelo fator alocação.

Experimento	**Média do tempo (s)**	**Intervalo de confiança (s)**
1	0.45796	(0.45567, 0.46025)
2	0.45723	(0.45544, 0.45902)
3	0.41892	(0.41656, 0.42128)
4	0.41976	(0.41754, 0.42198)
5	0.45838	(0.45624, 0.46052)
6	0.454109	(0.4537, 0.45451)

4.2 Comparação dos eventos

Percebe-se que nos experimentos 2, 4 e 6, com alocação dinâmica, houve um aumento bastante perceptível de *L1-dcache-loads-misses* comparado aos experimentos 1, 3 e 5, com alocação estática, que não deve ser surpresa, pois a região da *stack* da memória deve ser mais compatível com as *caches*. Além disso, quando mantemos o tipo de alocação constante, nota-se que a otimização de *loop unrolling* tem um efeito de diminuição de tal evento.

Nota-se que nos experimentos 3 e 4, com otimização *loop interchange*, o número de *L1-dcache-loads-misses* foi consideravlemnte mais alto em comparação com os experimentos que utilizaram outras otimizações com a mesma alocação. Ou seja, a alocação teve o maior efeito sobre tal evento, porém dentro do conjunto de experimentos com a mesma alocação, a otimização *loop interchange* aumenta a ocorrência desses evntos, enquanto a otimização *loop unrolling* as diminui.

Nos experimentos 5 e 6, com otimização *loop unrolling*, houve uma diminuição de *L1-dcache-loads* e *branch-instructions* em relação aos experimentos 1, 2, 3 e 4, que apresentaram valores similares entre si para esses eventos, também como esperado. Entretanto, o valor de *branch-misses*, similar nos experimentos 1, 2, 3 e 4, aumenta quase 20% nos experimentos 5 e 6.

4.3 Influência dos fatores

Percebe-se que na análise 1 e 2, com variável resposta *L1-dcache-loads* e *L1-dcache-loads-misses*, o fator alocação teve muito mais influência do que o fator otimização. O oposto aconteceu para as análises 3 e 4, com variável resposta *branch-instructions* e *branch-misses*, onde a otimização teve muito mais influência que a alocação.

Análise	**Alocação**	**Otimização**	**Combinada**
1	0.749425232372584	0.0998220284573909	0.150752739170025
2	0.91234674236662	0.086274548555977	0.0013787090774027 4
3	6.87729099568712e-09	0.999999993013192	1.09516549136072e-10
4	0.00546033775752893	0.990755120362726	0.00378454187974538