Sistemas Distribuídos – UFRJ

Trabalho Prático 1

Aluno: Paulo Victor Innocencio

DRE: 116213599

Disciplina: Sistemas Distribuídos

(COS470)

Professor: Daniel Ratton Figueiredo

Introdução

O intuito deste relatório é mostrar as decisões feitas para o projeto e o passo a passo das partes cruciais dos programas. A linguagem utilizada para desenvolvimento foi a linguagem C++ e o ambiente de trabalho foi o Linux Ubuntu.

Somador com Spinlocks

O programa deveria calcular a soma de uma grande quantidade de números. A utilização de threads seria necessária para uma maior eficácia, visando reduzir o tempo de execução. Os números que deveriam ser somados estariam armazenados em um vetor de tamanho N e seus valores seriam números aleatórios no intervalo [-100; 100]. O parâmetro K seria o número de threads que iriam trabalhar em conjunto. Uma coordenação para controlar o acesso à região crítica se mostrou totalmente necessária. A estrutura Spinlock é uma primitiva de sincronização de nível inferior com exclusão mútua que gira enquanto aguarda adquirir um bloqueio. A mesma foi implementada da seguinte forma:

```
class Spinlock{

    std::atomic_flag locked = ATOMIC_FLAG_INIT;
public:
    void aquire()
    {
        while(locked.test_and_set()){}
    }
    void release()
    {
        locked.clear();
    }
};
```

Como sugerido pelo enunciado do problema, uma boa opção seria dividir a soma em K parcelas com N/K valores e deixar cada thread realizar a soma de cada parcela, somando o resultado em um acumulador comum, compartilhado. Essa medida foi implementada da seguinte forma:

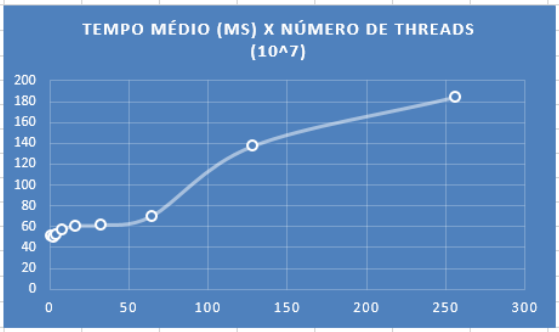
```
for (long int i = 0; i < N; i++){
    valores[i] = char(rand()%201 - 100);    //Obedecendo o intervalo [-100,100]
}

parcela = trunc(N/K);
float resto = N % K;
if(resto != 0){
    parcela +=1;
}
```

Após realizar todos os estudos de casos com os parâmetros requeridos pelos enunciados ($N = 10^7$, $N = 10^8$, $N = 10^9$ e $K = \{1, 2, 4, 8, 16, 32, 64, 128, 256\}$) chegamos aos seguintes resultados:

$N = 10^7$

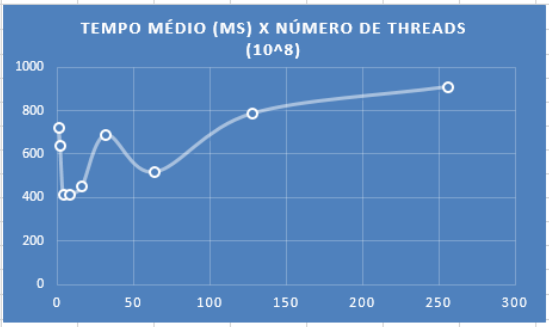
K	Tempo Médio (mS)
1	51,2
2	50,3
4	53,1
8	57,2
16	60,8
32	61,6
64	70
128	137,5
256	184,7



Podemos observar um comportamento de crescimento proporcional em relação a quantidade de threads e o tempo médio gasto.

$N = 10^8$

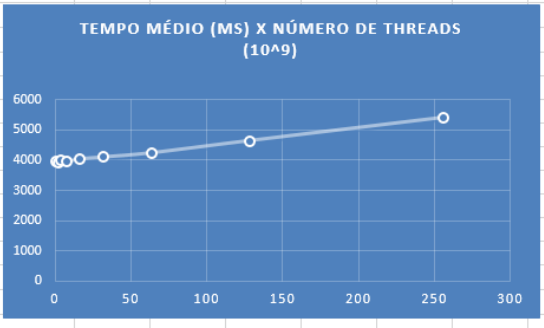
K	Tempo Médio (mS)
1	722,5
2	636,1
4	410,4
8	415,8
16	449,2
32	689,6
64	519,4
128	786,7
256	908



Podemos observar uma maior variação no comportamento. O aumento do número de threads operantes no programa mostrou ser vantajoso até certo ponto, sendo que após esse resultado ótimo, esse mesmo aumento começou a gerar uma carga de tempo final maior.

$N = 10^9$

K	Tempo Médio (mS)
1	3975,9
2	3936,2
4	3996,5
8	3954,3
16	4037,1
32	4104,2
64	4245,4
128	4643,8
256	5413,6



Nesse caso, o comportamento mostrou-se mais estável e proporcional.

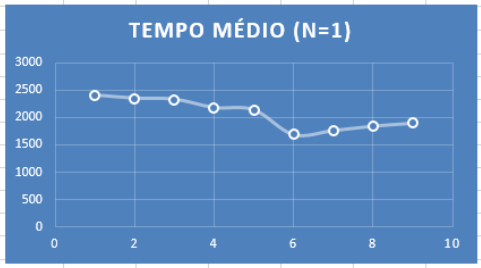
Produtor Consumidor com Semáforos

O clássico problema onde dois processos compartilham um buffer de tamanho fixo e o processo produtor insere informação no buffer, enquanto o processo consumidor remove informação do mesmo buffer compartilhado. Foi necessário implementar um mecanismo de coordenação para alternar o acesso das threads a região crítica.

Após realizar todos os estudos de casos com os parâmetros requeridos pelos enunciados N = {1, 2, 4, 16, 32} e (N_p, N_c) = {(1, 1), (1, 2), (1, 4), (1, 8), (1, 16), (2, 1), (4, 1), (8, 1), (16, 1)} chegamos aos seguintes resultados:

N = 1

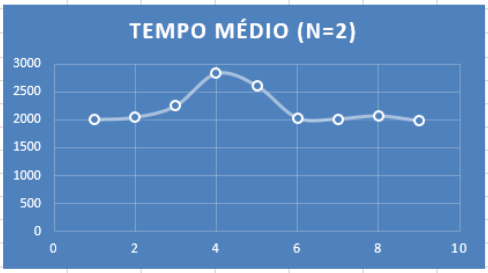
	N =1	
Ponto	(Np, Nc)	Tempo Médio (mS)
1	(1,1)	2401,666667
2	(1,2)	2345,444444
3	(1,4)	2327
4	(1,8)	2178,222222
5	(1,16)	2137,777778
6	(2,1)	1694,222222
7	(4,1)	1765,333333
8	(8,1)	1843,555556
9	(16,1)	1896,555556



Par (N_p, N_c) de melhor desempenho: (2,1)

N = 2

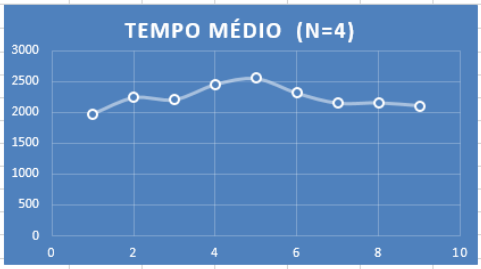
	N = 2	
Ponto	(Np, Nc)	Tempo Médio (mS)
1	(1,1)	1998,444444
2	(1,2)	2041,555556
3	(1,4)	2248,444444
4	(1,8)	2833,222222
5	(1,16)	2612,111111
6	(2,1)	2030,111111
7	(4,1)	2013,555556
8	(8,1)	2067,444444
9	(16,1)	1981,666667



Par (N_p, N_c) de melhor desempenho: (16,1)

N = 4

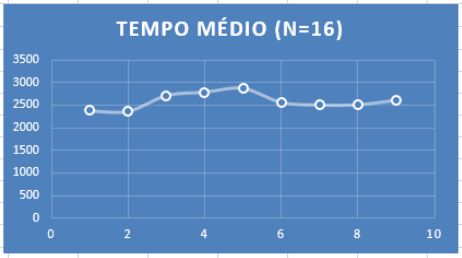
	N = 4	
Ponto	(Np, Nc)	Tempo Médio (mS)
1	(1,1)	1973,555556
2	(1,2)	2239,111111
3	(1,4)	2205,777778
4	(1,8)	2457,111111
5	(1,16)	2554,222222
6	(2,1)	2312,888889
7	(4,1)	2147,555556
8	(8,1)	2154,666667
9	(16,1)	2106,333333



Par (N_p, N_c) de melhor desempenho: (1,1)

N = 16

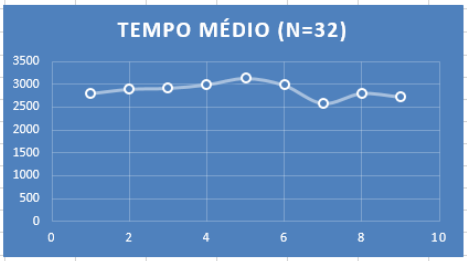
	N = 16	
Ponto	(Np, Nc)	Tempo Médio (mS)
1	(1,1)	2378,777778
2	(1,2)	2370,888889
3	(1,4)	2712,111111
4	(1,8)	2784
5	(1,16)	2867,333333
6	(2,1)	2564,555556
7	(4,1)	2507,777778
8	(8,1)	2512,555556
9	(16,1)	2609,333333



Par (N_p, N_c) de melhor desempenho: (1,2)

N = 32

	N = 32	
Ponto	(Np, Nc)	Tempo Médio (mS)
1	(1,1)	2802,444444
2	(1,2)	2898
3	(1,4)	2919,222222
4	(1,8)	2995,333333
5	(1,16)	3134
6	(2,1)	2984,444444
7	(4,1)	2586,222222
8	(8,1)	2804,333333
9	(16,1)	2729,555556



Par (N_p, N_c) de melhor desempenho: (4,1)

Considerações finais

- A imagem de capa não é de minha autoria e foi retirada da internet
- Todos os códigos estão disponíveis em: https://drive.google.com/drive/folders/1jU7L9n7SUibS2ftHeR5FVrUIedj3JA9G?usp=sharing