Jean Fobe, 7630573

INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
UNIVERSIDADE DE SÃO PAULO, 2018/05/04
Computação Paralela e Distribuída - MAC5742/MAC0219 2018-1
Prof. Alfredo Goldman

Exercício de Programação: Gestão de seção crítica

- Pelos programas fornecidos, entendeu-se que os algorítmos bakery e gate foram implementados de forma a demonstrar o seu funcionamento. Para tanto são colocados processos por meio de threads que fazem acessos a uma região crítica um número de vezes especificadas (30), tirando a média e o desvio padrão.

- Para observar os efeitos da contenção fiz um script adicional 'this.sh', que rodará o programa 'main' com valores de potências de 2 para o argumento do número de threads, indo de 2^0 a 2^{13}, e argumento de tempo total com valores (potências de 2) de 2^{16} a 2^{32}. Modifiquei também os arquivos fornecidos para imprimir os resultados das execuções em somente uma linha, facilitando a análise posterior. Os resultados obtidos foram colocados em um arquivo
'ep4-data.csv' com colunas 'Algorithm, Execution_Number, Elapsed_Time_ns, Access_Avg, Access_StdDev, Total_Time, Num_Threads'.

- Com isso, conforme o gráfico 1, podemos observar o comportamento dos algorítmos em função do número de threads sendo controladas e o tempo total de execução. Não foi possível aguardar o termino do meu script ‘this.sh’, pois deixei o computador rodando por mais de 10 horas sem atingir o final da execução.

- Os resultados obtidos foram:

- Gate teve medias e desvios menores para um número de threads menor que 16, mas apresentou maiores índices de starvation.
- Infelizmente, não pude esperar a execução finalizar e com isso não observei efeitos de um número de threads maior que 32, mas estimo com o que foi observado, que os padrões de fairness serão mantidos com perda de performance proporcional.
- O aumento do número de threads melhorou a performance nos dois algoritmos, sendo o ‘bakery’ o com melhores resultados.
- O sumário das variáveis obtidas no arquivo ' .csv' foram:

```
 Algorithm    Execution_Number              Elapsed_Time_ns   Access_Avg     Access_StdDev     Total_Time     Num_Threads
bakery:2956   Min.   : 1.00    4069981311.000000      :   2   Min.   : 0.000   Min.   : 0.00   Min.   :16.00   Min.   :0.000
gate  :2940   1st Qu.: 8.00    1000135020335545.250000:   1   1st Qu.: 7.719   1st Qu.:10.07   1st Qu.:20.00   1st Qu.:1.000
              Median :15.00    100045701311.687500    :   1   Median :10.528   Median :11.12   Median :24.00   Median :2.000
              Mean   :15.48    1000847907335545.250000:   1   Mean   : 8.474   Mean   :10.63   Mean   :23.75   Mean   :2.405
              3rd Qu.:23.00    100157126920972.500000 :   1   3rd Qu.:11.449   3rd Qu.:11.70   3rd Qu.:28.00   3rd Qu.:4.000
              Max.   :30.00    1001992971335545.250000:   1   Max.   :12.033   Max.   :12.15   Max.   :32.00   Max.   :5.000
                               (Other)                :5889
```

Gráficos:

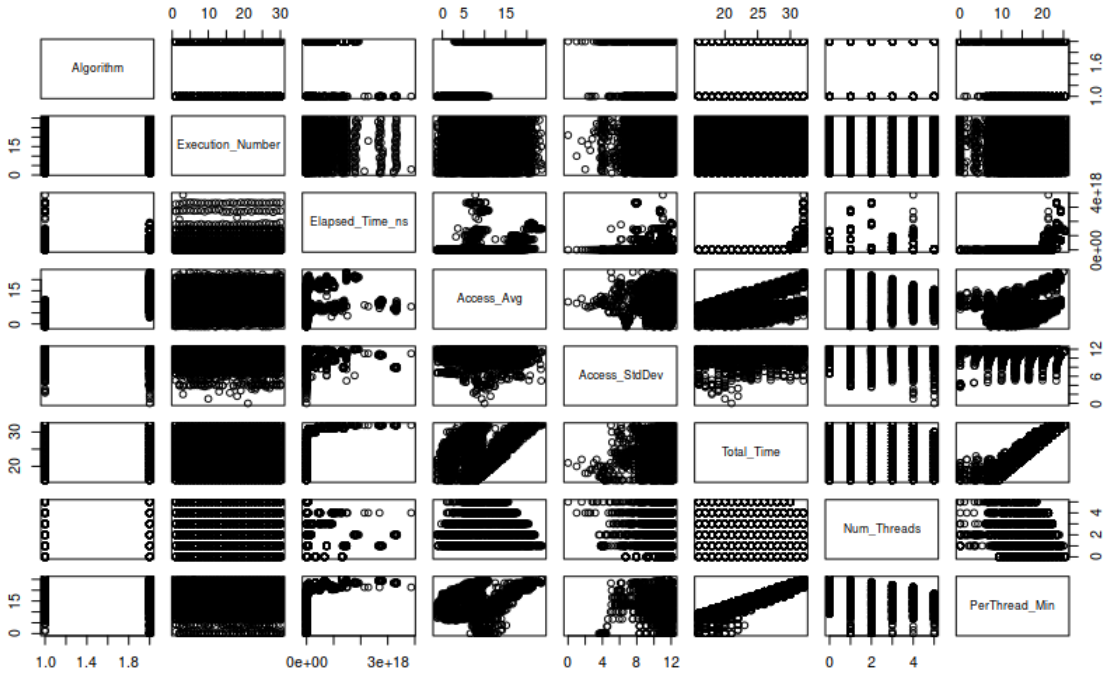

Gráfico 1: Relação entre variáveis duas-a-duas, das informações obtidas. ‘Access_Avg, Access_StdDev, Num_Threads, Total_Time, Per_Thread_Min’ em logarítmo de base 2 (note aqui como diferem as relações das médias e desvio padrão para cada algorítmo).

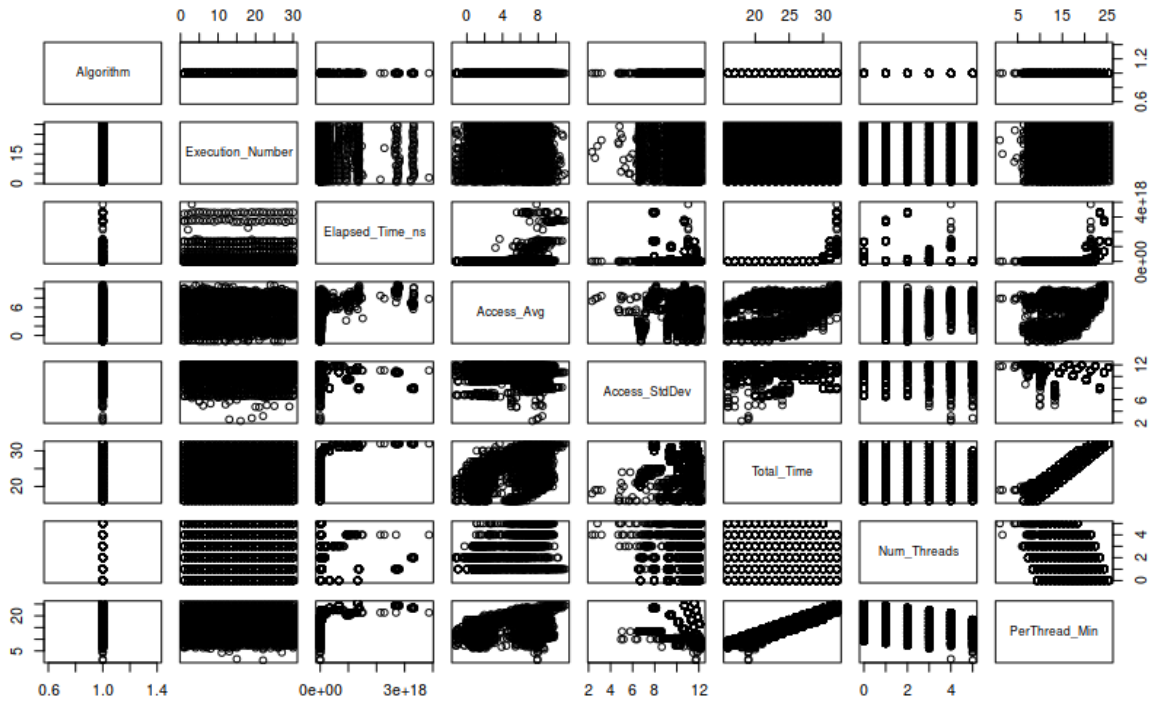

Gráfico 2: Semelhante ao acima, mas com dados somente do algoritmo ‘bakery’.

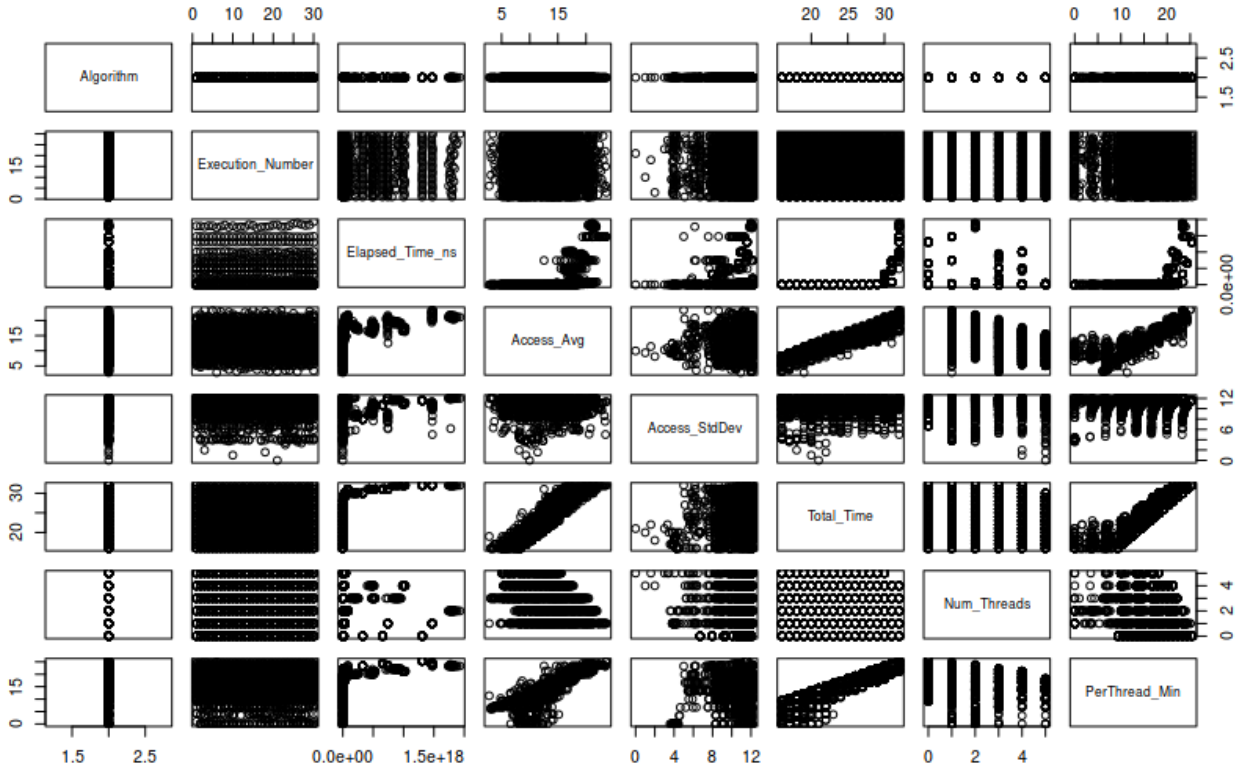

Gráfico 3: Semelhante ao acima, mas com dados somente do algoritmo ‘gate’. Observe a maior starvation com ‘gate’ pelo padrão entre ‘Total_time’, ‘PerThread_Min’ e ‘Access_Avg’.

Apendice:

- this.sh

```
#!/usr/bin/bash

echo "Algorithm,Execution_Number,Elapsed_Time_ns,Access_Avg,Access_StdDev,PerThread_Access_Count" >
ep4-data.csv;

for i in $(seq 0 13); do
        for j in $(seq 16 32); do
                ./main $((2**$i)) $((2**$j))  >> ep4-data.csv;

        done;
done;
```