Jean Fobe, 7630573

INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
UNIVERSIDADE DE SÃO PAULO, 2018/04/13
Computação Paralela e Distribuída - MAC5742/MAC0219 2018-1
Prof. Alfredo Goldman

Exercício de Programação: Contenção

- Pelo script fornecido, 'contentions.sh' entende-se que será compilado e executado um programa em C (Standard C99), usando OpenMP para paralelizar operações feitas em um vetor. O tamanho do vetor é especificado como argumento do script, bem como o número de threads. Outro parâmetro da execução é estático: a quantidade de condicionais, 'if's, que são colocados antes da região crítica do problema. O script varia a quantidade de 'if's entre 0 e 9 para os parâmetros fornecidos na entrada.

- Para observar os efeitos da contenção fiz um script adicional 'loopy.sh', que rodará o 'contentions.sh' com valores de potências de 2 para o argumento do vetor, indo de 2^0 a 2^{30}, e argumento de threads com valores (potências de 2) de 2^0 a 2^{13}. Modifiquei também o script 'contentions.sh', para receber potências de 2 de 2^0 a 2^{10}. Os resultados obtidos foram colocados em um arquivo 'contentions_out.csv' com colunas 'ARRAY_SIZE, NUM_THREADS, NUM_IFS, AVG5_TIME(s)' .

- Com isso, conforme o gráfico 1, podemos notar a influência do número de threads 'ifs', e o tamanho do array na performance do programa compilado 'test' pelas médias de tempo na execução. Não foi possível executar o script para tamanhos de vetor maiores que 2^{30}, pois a memória entrou no Swap do PC, piorando exponencialmente a performance.

- Os resultados obtidos foram:
    - Vetores menores contribuiram com perdas proporcionais ao seu tamanho (~2^n).
    - Infelizmente, talvez devido a arquitetura do processador do meu computador (AMD FX(tm)-8350 Eight-Core Processor), não foi observado o efeito de otimização esperado dos 'if's encadeados, só houve melhoria de performance expressiva entre o caso com apenas 1 'if'. Curiosamente, também não houve perda de performance para o acréscimo de dois ou mais 'if's.
    - O aumento do número de threads melhorou a performance para vetores de tamanhos maiores que 2^{10} (sizeof int).
    - O summario das variáveis obtidas no arquivo ' .csv' foram:

```
  ARRAY_SIZE         NUM_THREADS      NUM_IFS        AVG5_TIME(s)          index
Min.   :1.000e+00   Min.   :   1   Min.   :   1.0   Min.   :0.0000000   Min.   :   1
1st Qu.:1.280e+02   1st Qu.:   8   1st Qu.:   4.0   1st Qu.:0.0002412   1st Qu.:1194
Median :3.277e+04   Median :  96   Median :  32.0   Median :0.0024165   Median :2388
Mean   :6.927e+07   Mean   :1170   Mean   : 186.1   Mean   :0.0501923   Mean   :2388
3rd Qu.:8.389e+06   3rd Qu.:1024   3rd Qu.: 256.0   3rd Qu.:0.0220428   3rd Qu.:3581
Max.   :1.074e+09   Max.   :8192   Max.   :1024.0   Max.   :2.8062600   Max.   :4774
```

Gráficos:

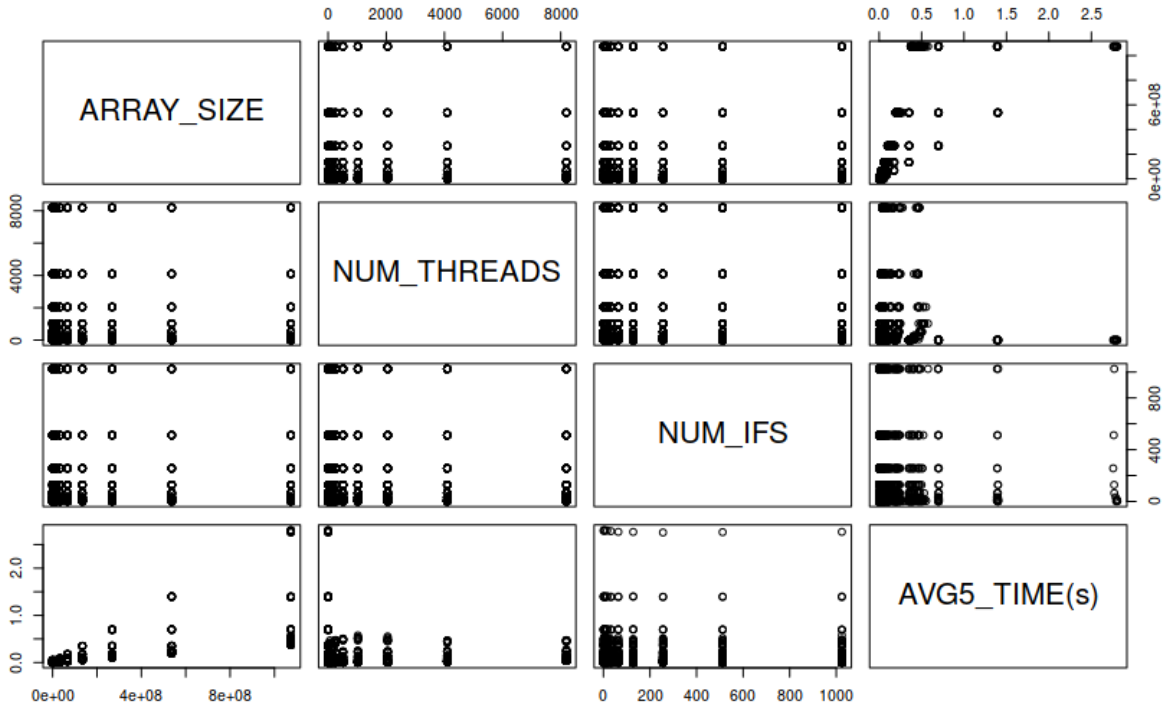

Gráfico 1: Relação entre variáveis duas-a-duas, das informações obtidas da execução do programa (note aqui as relações com 'AVG5_TIME(s)' das médias de tempo obtidas em segundos).

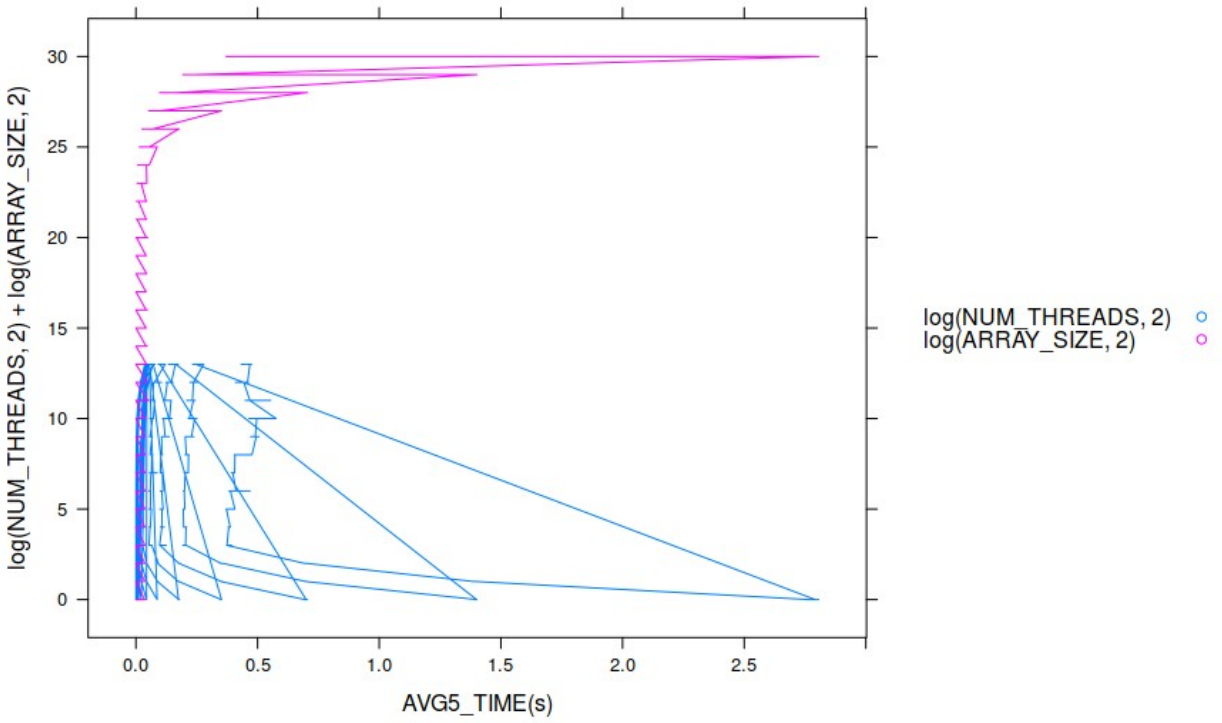

Gráfico 2: Relação entre médias de tempo, números de threads e tamanho do vetor. Note a perda de performance para um número baixo de threads.

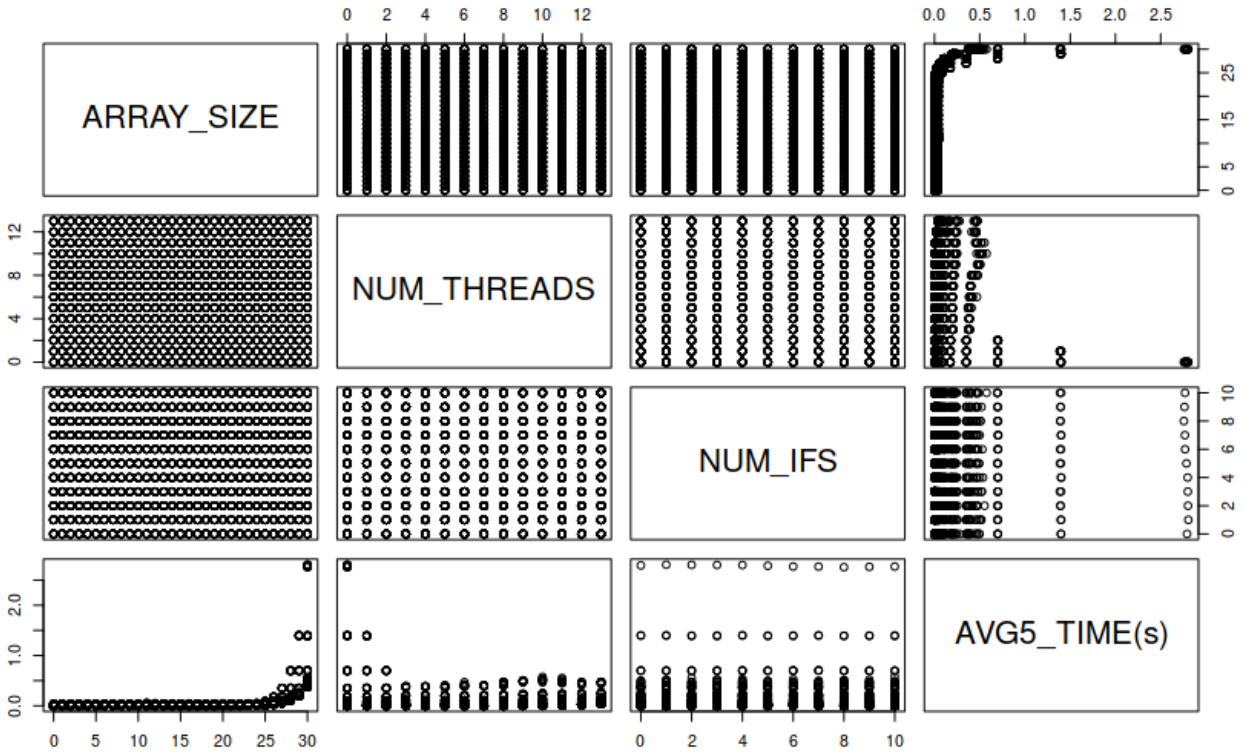

Gráfico 3: Valores de números de threads, tamanho do vetor e número de 'ifs' em logarítmo de base 2. Note de forma mais explicita o efeito não expressivo da quantidade de 'ifs'.

Apendice:

- loopy.sh

```
#!/bin/bash

RED='\033[0;31m';
GRN='\033[1;32m';
YEL='\033[1;33m';
BLU='\033[1;34m';
NC='\033[0m';

## Initialize .csv header & destroy/create file
echo "SIZE_VECTOR, NUM_THREADS, NUM_IFS, AVG5_TIME(s)" > contention_out.csv;

## Outer loop, iterate over array sizes up to 2^30;
for j in $(seq 0 30) ; do

       ## Inner loop, iterate over number of threads up to 4096 (limit per process is about ~>7000);
       for k in $(seq 0 13) ; do
              ## Check progress -- in colors !! --
              printf "\t\t${YEL}Execution: ${GRN}$j ${BLU}$k ${NC}\n";
              ## Run iteration, append .csv
              ./contention.sh $((2**$j)) $((2**$k)) 10 >> contention_out.csv;
       done;
done;
```

- diff -u contention.sh

```
@@ -7,7 +7,7 @@

 #define IF if (v[i] > max)

-#define NUM_EXEC 5
+#define NUM_EXEC 8

 static void populate_vector(size_t N, int v[])
 {
@@ -92,7 +92,8 @@
     for (i = 0; i < NUM_EXEC; ++i)
         times[i] = contention_test(N, T, vector);

-    fprintf(stdout, \" Average of %d executions: %lf s\\n\", NUM_EXEC, avg(NUM_EXEC, times));
+    //fprintf(stdout, \" Average of %d executions: %lf s\\n\", NUM_EXEC, avg(NUM_EXEC, times));
+    fprintf(stdout, \"%lf \\n\", avg(NUM_EXEC, times));

     free(vector);
     return 0;
@@ -103,6 +104,7 @@

 SIZE_VECTOR=$1
 NUM_THREADS=$2
+MAX_POW2IF=$3

 generate_ifs() {
     c_ifs=""
@@ -119,9 +121,11 @@

 run_for_if() {
     #for ((num_ifs=0; num_ifs<9; num_ifs++)); do
-    for num_ifs in $(seq 0 9); do
-        echo "Number of ifs: $num_ifs"
-        generate_c $num_ifs "temp.c"
+    for num_ifs in $(seq 0 $MAX_POW2IF); do
+          #echo -n "$1, $2, $((2 **$num_ifs)), "
+          echo -n "$1, $2, $num_ifs, "
+        #generate_c $num_ifs "temp.c"
+      generate_c $((2 ** $num_ifs)) "temp.c"
         gcc -Wall -O0 -std=c99 -fopenmp -o temp temp.c
         ./temp $1 $2
     done
```