Relatório EP1

André Luiz Abdalla 8030353
Mauricio Luiz Abreu Cardoso 6796479

20 de maio de 2018

Introdução

O algoritmo utilizado para a multiplicação das matrizes é o mais básico e simples para se resolver tal problema, dado que não queríamos inserir dificuldades desnecessárias

```
1  double** multiplicaMatrizes(double **mat_a, double **mat_b, int la, int ca, int cb) {
2      int c, d, k;
3      double sum = 0;
4
5      double **mat_c = malloc (ca * sizeof(double *));
6      for (c = 0; c < la; c++)
7          mat_c[c] = malloc (cb * sizeof(double));
8
9
10     for (c = 0; c < la; c++) {
11         for (d = 0; d < cb; d++) {
12             for (k = 0; k < ca; k++) {
13                 sum = sum + mat_a[c][k]*mat_b[k][d];
14             }
15             mat_c[c][d] = sum;
16             sum = 0;
17         }
18     }
19
20     return mat_c;
21 }
```

Listing 1: Função para multiplicar matrizes

Tendo a ferramenta multiplicadora e o que demais foi necessário, cuidamos de como paralelizar

Pthreads A multiplicação de matrizes foi paralelizada de acordo com o número de núcleos disponíveis. Se a matriz A tiver mais linhas do que o processador tiver cores, serão alocadas threads em mesma quantidade que o número de cores, e as multiplocações das linhas da matriz A serão divididas entre as threads, de modo igual. Se o resto da divisão de linhas da matriz A pelo número de cores for diferente de 0, a última thread terá menos linhas a serem multiplicadas. Vale ressaltar que no caso que o número de linha de A é menor que o número de cores, a quantidade de threads será igual a quantidade de linhas, e cada thread receberá somente uma linha.

Open MP Para o programa usando essa ferramenta, incluímos uma linha [1]. Além de garantir o paralelismo para o for mais externo, permite que determinemos quantas threads haverão (4 neste caso) e que ordenemos sua execução

[1] `#pragma omp parallel for schedule(static, 2) num_threads(4) ordered`