Enunciado do trabalho prático 2

Objetivo

Esta fase do trabalho prático tem como objetivo avaliar capacidade dos alunos para desenvolver programas que explorem paralelismo tendo como principal objetivo a redução do tempo execução do programa.

Introdução

Neste trabalho prático os alunos deverão utilizar o código desenvolvido no TP1 e, através de primitivas *OpenMP*, implementar uma versão do programa que em uma ou mais fases do algoritmo divida o processamento por vários fios de execução. Pretende-se que os alunos se foquem na **metodologia de desenvolvimento** de programas paralelos. Para tal, sugere-se a realização das seguintes tarefas:

1. Identificar quais os blocos de código com maior carga computacional;
2. Apresentar e analisar diferentes alternativas para exploração de paralelismo para cada bloco identificado em 1;
3. Selecionar a alternativa mais viável justificando com uma análise de escalabilidade
4. Implementação da versão paralela mais adequada ao ambiente de execução (nó do *Search* da fila *cpar*)
5. Analisar o desempenho da proposta implementada

Resultados a apresentar

Na submissão do trabalho deverão incluir os tempos de execução obtidos no nó do *Search*, usando 10 000 000 amostras (pontos no espaço X,Y representados por dois valores do tipo *float*) para 4 e 32 *clusters*. Na avaliação do trabalho serão também usados os tempos de execução para um número de *clusters* não divulgado. Nesta fase pretende-se que critério de paragem seja o número de iterações que deverá ser fixado a **20 iterações**.

Grupos, estrutura do código/relatório e datas

Os grupos deverão ser mantidos com a mesma constituição definida no primeiro trabalho prático.
O código e respetiva `Makefile` têm que *cumprir as regras* definidas no Anexo I, **o não cumprimento das regras definidas poderá originar uma penalização na avaliação**.
O relatório será em formato *pdf*, com um máximo de 2 páginas, excluindo anexos (os anexos só serão consultados se o docente achar relevantes). Deverão usar o *template* IEEE (em https://www.ieee.org/conferences/publishing/templates.html).
A submissão deverá consistir num ficheiro *zip* que quando descompactado deverá conter uma diretoria cujo nome é a constituição do grupo (exemplo *a43000_pg54000*), e dentro desta diretoria deve estar o relatório, a `Makefile` e as pastas com o programa indicadas no ponto 3 do Anexo I.
O ficheiro *zip* deverá ser submetido na plataforma *elearning* até às **23:59 horas do dia 18-Nov-22**.
A defesa deste trabalho será realizada juntamente com a entrega do TP3, em Janeiro de 2023. Nessa defesa poderá ser revista a nota das primeiras entregas.

Critérios de avaliação

A nota deste trabalho, terá em consideração: **(i)** proposta de exploração de paralelismo e sua implementação em *OpenMP* incluindo a legibilidade do código **(40%); (ii)** desempenho da versão paralela do código desenvolvido, será contabilizado o tempo com o número de fios de execução indicados na `Makefile` **(20%)**; **(iii)** relatório (análise de escalabilidade, medição do perfil de execução, balanceamento de carga...) **(40%).**

Anexo I - Estrutura do código e *Makefile* a submeter

1. Input dos valores dos pontos e dos *clusters*

O input será novamente gerado através da função desenvolvida para a primeira fase do trabalho, sendo que nesta fase o número de pontos e amostras devem ser passados como parâmetros. Assim sendo o programa deve receber 3 parâmetros de entrada: 1º numero de pontos, 2º número de clusters e 3º número de fios de execução (exemplo: `./k_means 10000000 4 2`).

2. Validação do resultado e *output* do programa

Para validação do algoritmo deverão ser usados os resultados da primeira fase do trabalho prático, utilizando o critério de paragem de **20 iterações**. Neste caso, o *output* do programa deverá ser aproximadamente (`gcc 7.2.0` no cluster *Search*, noutras máquinas o resultado pode ser diferente) o seguinte (N-número de prontos, K-número de *clusters*):

```
N = 10000000, K = 4
Center: (0.250, 0.750) : Size: 2498728
Center: (0.250, 0.250) : Size: 2501731
Center: (0.750, 0.250) : Size: 2499396
Center: (0.750, 0.750) : Size: 2500145
Iterations: 20
```

3. Organização do código e *Makefile*

Deverá ser incluído um ficheiro com o nome `Makefile` que gere um executável com o nome `k_means` numa subdiretoria `bin`. Os ficheiros fonte devem estar localizados na subdiretoria `src`. O programa será executado utilizando os comandos `make runseq` e `make runpar`. A variável de ambiente *CP_CLUSTERS* será usada para indicar o número de clusters a ser usado na execução do programa **(não deverá ser definida pelos alunos**, o programa correrá com diferentes números de clusters para o relatório deverá ser usado 4 e 32 clusters). Os alunos poderão indicar as *CFLAGS* que entenderem, desde suportadas pelo `gcc 7.2.0` disponível no *Search*.
Na `Makefile` deverá ser definido o número de *threads* ideais.

Exemplo:

```
CC       = gcc
BIN      = bin/
SRC      = src/
INCLUDES = include/
EXEC     = k_means
THREADS  = ...

CFLAGS = ...

.DEFAULT_GOAL = k_means

k_means: $(SRC)k_means.c $(BIN)utils.o
 $(CC) $(CFLAGS) $(SRC)k_means.c $(BIN)utils.o -o $(BIN)$(EXEC)

$(BIN)utils.o: $(SRC)utils.c $(INCLUDES)utils.h
 $(CC) $(CFLAGS) -c $(SRC)utils.c -o $(BIN)utils.o

clean:
 rm -r bin/*

runseq:
 ./$(BIN)$(EXEC) 10000000 $(CP_CLUSTERS)

runpar:
 ./$(BIN)$(EXEC) 10000000 $(CP_CLUSTERS) $(THREADS)
```