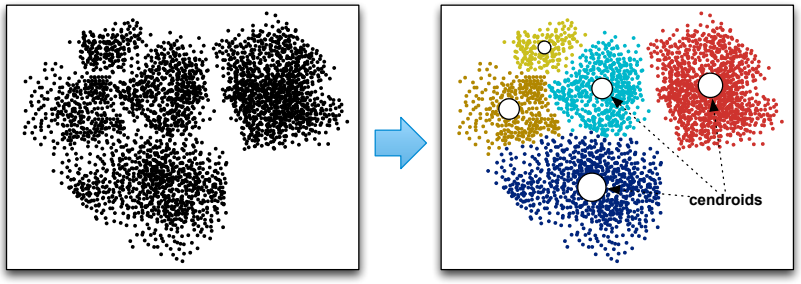
Trabalho 1 – Clusterização de Dados em Paralelo

INE5410 – Programação Concorrente – UFSC
Prof. Márcio Castro

1 Introdução

Análises baseadas em clusterização possuem aplicações em diferentes áreas do conhecimento científico, como mineração de dados, reconhecimento de padrões, análise de imagens e bioinformática. Nesse contexto, uma técnica de clusterização amplamente utilizada é o *k-means clustering*.

Formalmente, o *k-means clustering* pode ser definido da seguinte forma: dado um conjunto de n pontos em um espaço real d-dimensional, o problema consiste em particionar esses n pontos em k partições, de forma a minimizar a distância quadrática média de cada ponto ao centroide da partição ao qual pertence. A Figura 1 ilustra uma instância desse problema.



Diversas heurísticas foram propostas para endereçar o problema de *k-means clustering*. Dentre elas, uma das mais difundidas é o algoritmo de Lloyd's, também conhecido com algoritmo *k-means*. Essa heurística é capaz e encontrar uma solução que é mínimo local para o problema, baseando-se em uma estratégia iterativa.

A versão sequencial do algoritmo *k-means* é apresentada abaixo. A ideia é utilizar o conceito de distância euclidiana mínima para iterativamente particionar o conjunto de pontos. O algoritmo funciona da seguinte forma. Inicialmente o conjunto de pontos é distribuído de maneira aleatória entre os centroides. Depois, os pontos são reagrupados em novas partições considerando a distância euclidiana mínima entre eles e os centroides – pontos são atribuídos à partição mais próxima. Em seguida, o centroide de cada partição é recalculado, considerando a média de todos os pontos na partição, e todo o procedimento é repetido até que nenhum centroide mude e cada ponto esteja a uma distância maior do que a distância mínima de aceite.

Algorithm 1.1: K-MEANS(*partitions*, *points*)

```
procedure POPULATE(partitions, points)
 for each part ∈ partitions
   do part.points ← ∅
 for each pnt ∈ points
   do { part ← NEAREST_PARTITION(pnt, partitions)
      { part.points ∪ pnt

procedure COMPUTE_CENTROIDS(partitions)
 for each part ∈ partitions
   do part.centroid ← COMPUTE_MEAN(part.points)

main
 RANDOM_INIT(partitions, points)
 repeat
   { POPULATE(partitions, points)
   { COMPUTE_CENTROIDS(partitions)
 until HAS_CHANGED(partitions) and TOO_FAR(partitions)
 return (partitions)
```

2 Definição do Trabalho

O primeiro trabalho da disciplina de Programação Concorrente consiste em desenvolver uma versão paralela do *k-means*. A solução deverá utilizar a biblioteca **POSIX threads** para paralelizar o código. A versão paralela deverá receber um novo parâmetro de entrada referente ao número de *threads* a serem utilizadas na computação. A estratégia de paralelização a ser utilizada para dividir a computação entre as *threads* deverá ser escolhida pelos alunos.

Você deverá utilizar como base a versão sequencial do *k-means* implementada em C disponível no Moodle. Nessa versão, os parâmetros de entrada são: o número de dimensões do problema (`dimension`); um conjunto de pontos (`npoints`); o número de centroides (`ncentroids`); e a distância mínima aceitável (`mindistance`) entre cada ponto e um centroide; e a semente do gerador de números aleatórios a ser utilizada na fase de inicialização aleatória dos pontos nos centroides (`seed`).

A Figura 1 mostra o resultado da execução do *k-means* para 8 partições e 128 pontos distribuídos aleatoriamente em um ambiente bidimensional utilizando como distância mínima o valor 2.0. A semente do gerador de números aleatórios utilizada nesse exemplo foi 987. A saída do programa mostra quais pontos (identificados de 0 até `npoints`-1) pertencem a cada uma das partições.

```
./km 128 2 8 2.0 987

Partition 0:
7 16 35 54 72 79 83 93 102 103 105 106 110 115 118 126
Partition 1:
1 13 14 15 29 31 46 51 59 69 70 77 84 97 109 113 117 120
Partition 2:
8 10 19 34 40 48 56 62 64 73 75 81 82 85 89 95 98 100
Partition 3:
0 5 20 24 32 61 67 86 92 99 104 119 123 127
Partition 4:
4 23 26 27 37 45 50 52 57 63 71
Partition 5:
2 25 41 44 47 55 58 76 78 90 101 114 124
Partition 6:
9 12 17 21 22 28 33 42 43 53 65 66 68 74 87 88 96 108 111 116 121 122 125
Partition 7:
3 6 11 18 30 36 38 39 49 60 80 91 94 107 112
```

Figura 1: Resultado da execução do *k-means*.

3 Grupos, Avaliação e Entrega

O trabalho deverá ser realizado em grupos de até **2 alunos**. Os alunos serão responsáveis por formar os grupos com auxilio da ferramenta "**Escolha de Grupos - Trabalho 1 (T1)**" disponível no Moodle.

Pelo menos um dos integrantes de cada grupo deverá submeter um arquivo contendo o código fonte em C contendo a solução do trabalho através do Moodle. A data/hora limite para o envio dos trabalhos é **23/05/2016 às 23h55min**. **Não será permitida a entrega de trabalhos fora desse prazo: trabalhos não entregues no prazo receberão nota zero.**

Após a data limite para entrega, os alunos deverão apresentar o trabalho ao professor assim como mostrar sua solução em funcionamento. As apresentações serão feitas durante as aulas nos seguintes dias:

- **24/05/2016:** Grupos A ao G (demais grupos estão liberados desta aula e terão presença confirmada).
- **31/05/2016:** Grupos H ao O (demais grupos estão liberados desta aula e terão presença confirmada).

O professor irá avaliar não somente a corretude mas também o desempenho e a clareza da solução. Além disso, os alunos serão avaliados pela apresentação e entendimento do trabalho. **A implementação e apresentação valerão 40% e 60% da nota do trabalho, respectivamente**.