Trabalho Prático 2 - Programação Distribuída

Este trabalho prático tem como principal objetivo a utilização das capacidades de programação distribuída em Java através do modelo de Actores para um problema de cálculo computacional. Deve ser usada a biblioteca Akka para suporte ao modelo de Actores.

Descrição do Problema

Considere o problema de calcular o valor de PI, definido pela divisão do valor da circunferência pelo valor do diâmetro de um círculo. Existem vários algoritmos matemáticos que permitem estimar o valor de PI com base em cálculos. Considere o método de Monte-Carlo[1], e o método de Séries de Gregory-Leibniz[2].

O método de Monte-Carlo consiste em gerar aleatoriamente uma grande quantidade de pontos numa área de largura e altura igual a 1 (as coordenadas x e y variam entre 0 e 1). O valor de PI é o produto de 4 pela divisão entre a quantidade de números gerados com uma distância inferior a 1 unidade a partir da raíz (0,0) e a quantidade total de números gerados.

O método de Séries de Gregory-Leibniz é calculado através da fórmula:

$$\pi = 4\left[1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots\right] = 4\sum_{k=0}^{\infty}\frac{(-1)^k}{2k+1}.$$

Pode ser implementado com o seguinte algoritmo:

```
1        double factor = 1.0;
2        double sum = 0.0;
3        for (k = 0; k < n; k++) {
4           sum += factor/(2*k+1);
5           factor = -factor;
6        }
7        pi_approx = 4.0*sum;
```

Ao paralelizar este algoritmo, é necessário ter em atenção o potencial conflito na variável partilhada "factor", que deverá ser substituída pelo seguinte código:

```
1           sum += factor/(2*k+1);
2           factor = -factor;
```

by

```
1           if (k % 2 == 0)
2              factor = 1.0;
3           else
4              factor = -1.0;
5           sum += factor/(2*k+1);
```

[1] https://www.geeksforgeeks.org/estimating-value-pi-using-monte-carlo/
[2] https://www.mathscareers.org.uk/article/calculating-pi/

Implementação do Problema através do Método de Séries de Gregory-Leibniz (10 vals)

Desenvolva uma aplicação distribuída, usando o modelo de actores, para o cálculo de PI usando o método de Séries de Gregory-Leibniz. O objectivo desta implementação é modelar a distribuição do cálculo computacional pelos vários actores.

A aplicação recebe como entrada a quantidade de iterações *k* para estimar o valor final, assim como o número de cálculos concorrentes a suportar. Deve produzir como resultado o valor estimado de PI.

1) Apresente o tipo de actores que irá usar para a execução distribuída deste cálculo, assim como a sua quantidade.
2) Apresente o tipo de mensagens trocadas entre os actores, identificando os campos das mensagens, assim como a identificação do tipo de actores que envia e recebe estas mensagens.
    - Deve considerar a função *main* caso esta também envie/receba mensagens.
    - Sugestão: pode complementar esta descrição com um diagrama de sequência ou similar referindo os actores e as mensagens trocadas.
3) Descreva o comportamento dos actores, despoletado ao receber mensagens.
    - Será valorizada uma utilização eficiente de todos os actores disponíveis para aumentar o nível de concorrência e reduzir o código sequencial na aplicação.
4) Descreva o comportamento da função *main*, tendo em conta os parâmetros de entrada e o valor de saída.
    - Considere a opção da função *main* permitir ao utilizador especificar os valores dos parâmetros em *runtime* (ao executar a aplicação por oposição à sua compilação).

Implementação do Problema através do Método de Monte-Carlo (10 vals)

Desenvolva uma aplicação distribuída, usando o modelo de actores, para o cálculo de PI usando o método de Monte-Carlo. O objectivo desta implementação é modelar a distribuição do cálculo computacional pelos vários actores. A aplicação recebe como entrada a quantidade de pontos a gerar para estimar o valor final, assim como o número de cálculos concorrentes a suportar. Deve produzir como resultado o valor estimado de PI.

1) Apresente o tipo de actores que irá usar para a execução distribuída deste cálculo, assim como a sua quantidade.
2) Apresente o tipo de mensagens trocadas entre os actores identificando os campos das mensagens, assim como a identificação do tipo de actores que envia e recebe estas mensagens.
    - Deve considerar a função *main* caso esta também envie/receba mensagens.
    - Sugestão: pode complementar esta descrição com um diagrama de sequência ou similar referindo os actores e as mensagens trocadas.
3) Descreva o comportamento dos actores, despoletado ao receber mensagens.
    - Será valorizada uma utilização eficiente de todos os actores disponíveis para aumentar o nível de concorrência e reduzir o código sequencial na aplicação.
4) Descreva o comportamento da função *main*, tendo em conta os parâmetros de entrada e o valor de saída.
    - Considere a opção da função *main* permitir ao utilizador especificar os valores dos parâmetros em *runtime* (ao executar a aplicação por oposição à sua compilação).

Entrega

A entrega deste trabalho prático consistirá num relatório onde deve apresentar uma explicação para cada item anterior, assim como um extrato do código relevante para esse mesmo item. O relatório deverá estar no formato PDF.

As duas aplicações devem estar contidas, cada uma, num ficheiro de código fonte Java distinto.

Todos os ficheiros (relatório + código-fonte) devem ser enviados num arquivo compactado do tipo **ZIP**, com a identificação dos elementos do grupo, número e nome.