Laboratório de Programação Concorrente

Lab3
DNA Sequence - 25.2

Objetivo

Compreender a diferença entre execução **serial** e **concorrente** em um problema real de processamento de dados, **aplicando boas práticas de divisão de tarefas, sincronização e análise de desempenho**.

Neste laboratório, o aluno desenvolverá uma versão concorrente em **Java** para o problema de contagem de padrões em sequências de DNA, com base em uma implementação serial já fornecida.

Cenário Básico: Análise de Sequências de DNA

Imagine que você está desenvolvendo uma ferramenta bioinformática capaz de analisar grandes conjuntos de sequências genéticas. Cada sequência é uma cadeia composta pelos caracteres **A, C, G e T**.

O objetivo é contar quantas vezes um **padrão alvo** (por exemplo, `"CGTAA"`) aparece em todas as sequências armazenadas em um diretório com arquivos de texto contendo as sequências.

O problema:

- O diretório pode conter **centenas de arquivos** com várias linhas, cada uma representando uma sequência de DNA.
- O padrão pode aparecer **múltiplas vezes** dentro da mesma sequência.
- O desafio é **paralelizar** esse processo de contagem, aproveitando múltiplos núcleos de CPU e diminuindo o atraso causado pelas operações bloqueantes de IO.

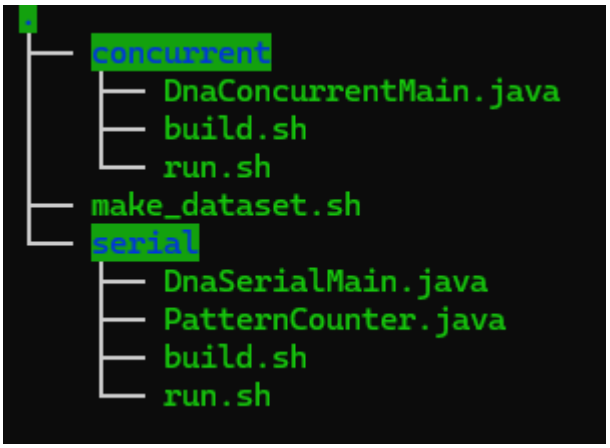
Visão geral do código base

No código base vocês encontrarão a implementação serial para este problema em Java. A partir de qual você pode construir sua solução concorrente.

A entrega, detalhada nas seções seguintes. Iremos avaliar tanto as possibilidades de plágio entre os alunos quanto a geração automática de código.

https://github.com/giovannifs/fpc/tree/master/2025.2/Lab3

O código está organizado na seguinte hierarquia:



Preparação

1. Clone o repositório do código base

   ```
   git clone [link do repositório]
   ```

Inicialmente, você deve usar o script `make_dataset.sh` para gerar uma base de dados para testes. Você pode especificar a quantidade de arquivos a serem gerados. Para isso, supondo que você queira gerar 150 arquivos, você deve executar o seguinte comando:

```
bash make_dataset.sh 150
```

Isso resultará na criação de um diretório dataset na raiz do projeto.

2. Execute a versão serial da solução. Para isso, você deve ir até o diretório serial e executar:

   ```
   bash build.sh
   ```

   ```
   bash run.sh <sequência_DNA>
   ```
   (por exemplo, GTAA)

O script `run.sh` executará automaticamente a versão serial considerando o diretório dataset como a raiz para as sequências de DNA, e, exibirá o tempo de execução (via `time`).

Desenvolvendo a Solução Concorrente

Nesta etapa, você deve implementar a versão concorrente do programa no diretório concurrent. O objetivo é dividir o trabalho, previamente executado por uma única thread, entre múltiplas threads, aproveitando melhor os núcleos do processador.

No código disponibilizado, a classe DnaConcurrentMain.java possui inicialmente uma cópia da solução serial. Você deve atualizar esta classe de forma a tornar esta solução de fato concorrente. Para isso, é importante pensar bem em questões como: Como devo dividir o trabalho entre as threads? Quantas threads devo criar? Como devo fazer a thread principal esperar o processamento das threads filhas? Como devo sincronizar os resultados das threads de forma segura?

Para executar a versão concorrente da solução, você deve ir até o diretório concurrent e executar:

```
bash build.sh
bash run.sh <sequência_DNA>
```
(por exemplo, GTAA)

Prazo

11/11/2025 às 16h00

Entrega

Você deve criar e manter um repositório privado no GitHub com a sua solução. No entanto, a entrega do laboratório deverá ser realizada por meio de submissão online utilizando o script submit-answer.sh, disponibilizado na estrutura de arquivos do próprio laboratório. Uma vez que você tenha concluído sua resposta, seguem as instruções:

1) Crie um arquivo lab3_matr1_matr2.tar.gz somente com o “src” do repositório que vocês trabalharam. Para isso, supondo que o diretório raiz de seu repositório privado chama-se lab3_pc, você deve executar:
   ```
   tar -cvzf lab3_matr1_matr2.tar.gz lab3_pc/src
   ```
2) Submeta o arquivo lab3_matr1_matr2.tar.gz usando o script submit-answer.sh, disponibilizado no mesmo repositório do laboratório:

```
bash submit-answer.sh lab3 path/lab3_matr1_matr2.tar.gz
```

Lembre-se que você deve manter o seu repositório privado no GitHub para fins de comprovação em caso de problema no empacotamento ou transmissão online. Alterações no código realizadas após o prazo de entrega não serão analisadas.

Exemplos de sintaxe em Java

Exemplo de classe de trabalho

```
public class Work implements Runnable {

    @Override

    public void run() {

        // coloque aqui a tarefa que a thread deve executar

    }

}
```

Exemplo de criação e uso das threads

```
// cria uma instância da classe de trabalho

Work work1 = new Work();

Work work2 = new Work();


// cria as threads associadas às tarefas

Thread t1 = new Thread(work1);

Thread t2 = new Thread(work2);


// inicia as threads

t1.start();
```

```
t2.start();


// espera as threads terminarem
try {
    t1.join();
    t2.join();
} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
}
```