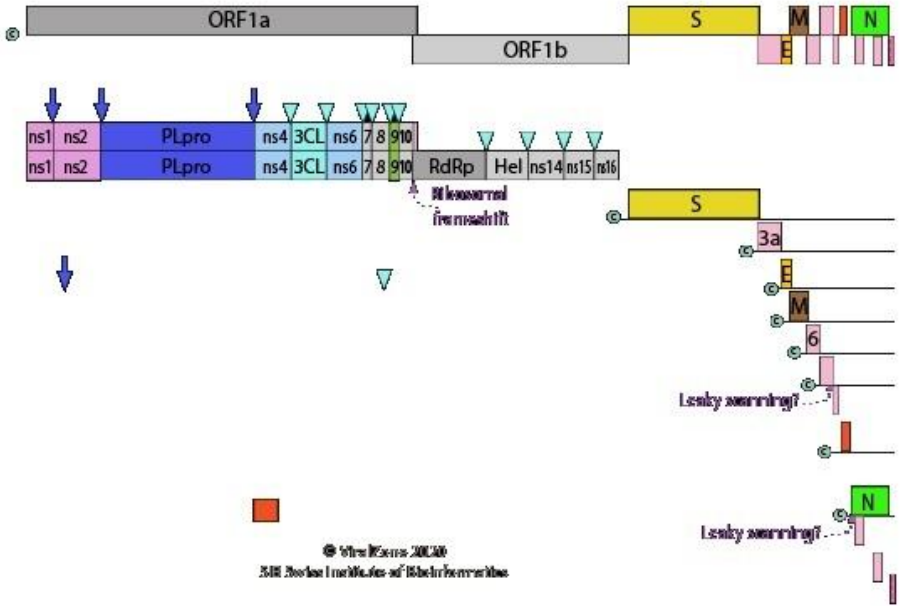
Universidade Fernando Pessoa

Sistemas Operativos

Trabalho Prático – Parte 1



Sistemas Operativos

Pedro Sobral

pmsobral@ufp.edu.pt

Bruno Gomes

bagomes@ufp.edu.pt

Março de 2021

Universidade Fernando Pessoa

Faculdade de Ciências e Tecnologias

Objectivo:

Criar um programa capaz de comparar eficientemente genomas do vírus SARS-CoV-2 na busca por mutações genéticas.

1. Definição do problema

Desde a descoberta do vírus SARS-CoV-2 e da divulgação do primeiro genoma de Whan (China), milhares de novas mutações têm vindo a ser detetadas pela comunidade científica. Uma vez que a comparação entre variantes de um vírus requer uma revisão integral aos genomas em análise, o problema tende a traduzir-se numa operação computacionalmente exigente. Com vista a otimizar o processo de comparação de genomas é proposta a implementação de uma solução para paralelização da verificação de mutações recorrendo a mecanismos de multiprocessamento. Dada a complexidade do problema, uma implementação base para a comparação (simplificada) entre genomas é fornecida, bem como o dataset de input, disponibilizado pela $NCBI^4$, com todos os genomas do vírus registados no continente europeu. Uma visão de alto nível sobre a estrutura do dataset é apresentada na figura 1.

Para esta primeira fase do projeto, o processo pai deverá carregar para memória todos os genomas (M genomas) presentes no dataset fornecido, lançando N filhos (M > N). Cada um dos processo filho parte de um ou mais genoma base, atribuídos pelo processo pai, verificando todas as mutações existentes para os restantes genomas. Assim que cada uma das C(M, 2) comparações é concluída, os processos filho deverão retornar ao pai os o resultado da comparação no seguinte formato:

#pid|genoma_base;genoma_a_comparar|gene$array_mutações

Após o lançamento dos N filhos, o processo pai deverá manter-se em escuta e recolher do canal de comunicação partilhado os resultados depositados por cada um dos processos

filho, escrevendo para ficheiro as sequências lidas. O programa termina com a impressão do tempo total decorrido desde o início da sua execução.

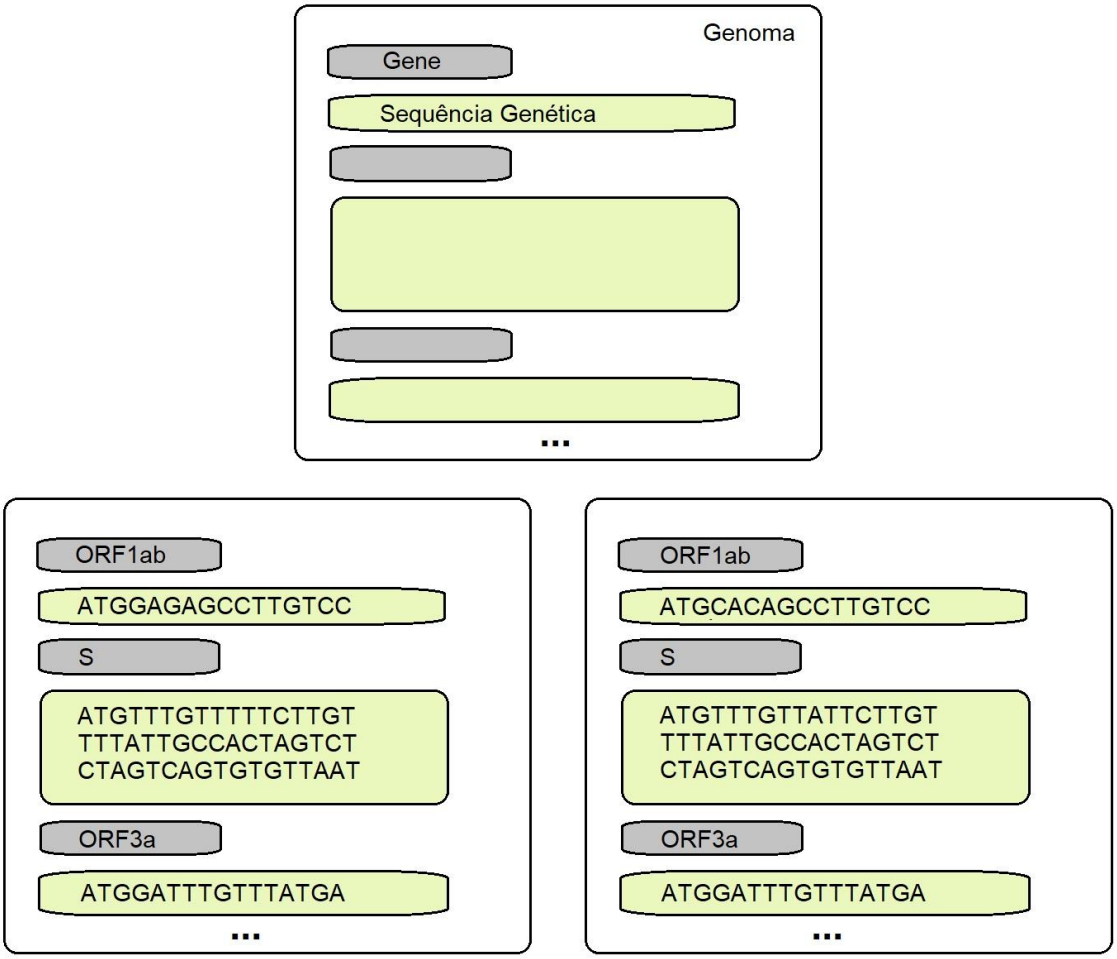


Figura 1 - Genoma, gene e sequência genética

2. Requisitos

Para a 1º fase de submissão serão considerados os seguintes requisitos:

A. (10%) Receber por parâmetro o número (N) de processos filhos a criar, balanceando de forma equitativa a carga computacional.

B. (20%) Lançar N filhos, cada um deles será responsável por partir dos vários genomas base definidos pelo processo pai e verificar a existência de mutações relativamente aos restantes genomas. Nesta fase nenhum canal de comunicação entre os processos filho e pai será necessário, devendo cada um dos filhos escrever para um ficheiro próprio o resultado da sua execução. Após verificação de cada genoma base cada processo filho deverá enviar um sinal ao pai *SIGURS1*[1] notificando que uma comparação adicional se encontra concluída. O pai deve esperar pela finalização dos seus filhos.

C. (10%) Com base no conhecimento da totalidade do problema, assim como do número de genomas base já comparados, o processo pai deverá imprimir periodicamente uma estimativa da % de execução do programa (0% = 0 genomas comprados; 100% = todos os genomas comparados).

D. (35%) Esta etapa implica que o programa tire partido da comunicação entre processos com recurso a *pipes*. Cada filho deve retornar as mutações resultantes para o pai através do *pipe*. O programa deve implementar um protocolo de comunicação que permita ao pai saber que genomas e genes foram comparados e as suas mutações, assim como o processo filho responsável pela comparação. O pai recebe as sequências resultantes, desserializa a mensagem em mutações e escreve o resultado para ficheiro. Nesta etapa o programa deve fazer recurso da função *readn* e *writen*[2]
    a. Protocolo: #pid|genoma_base;genoma_a_comparar|gene$array_mutações
    b. Nota: Com vista a reduzir o tamanho final do ficheiro resultante, o array de mutações poderá ser substituído pelo nº de mutações encontradas

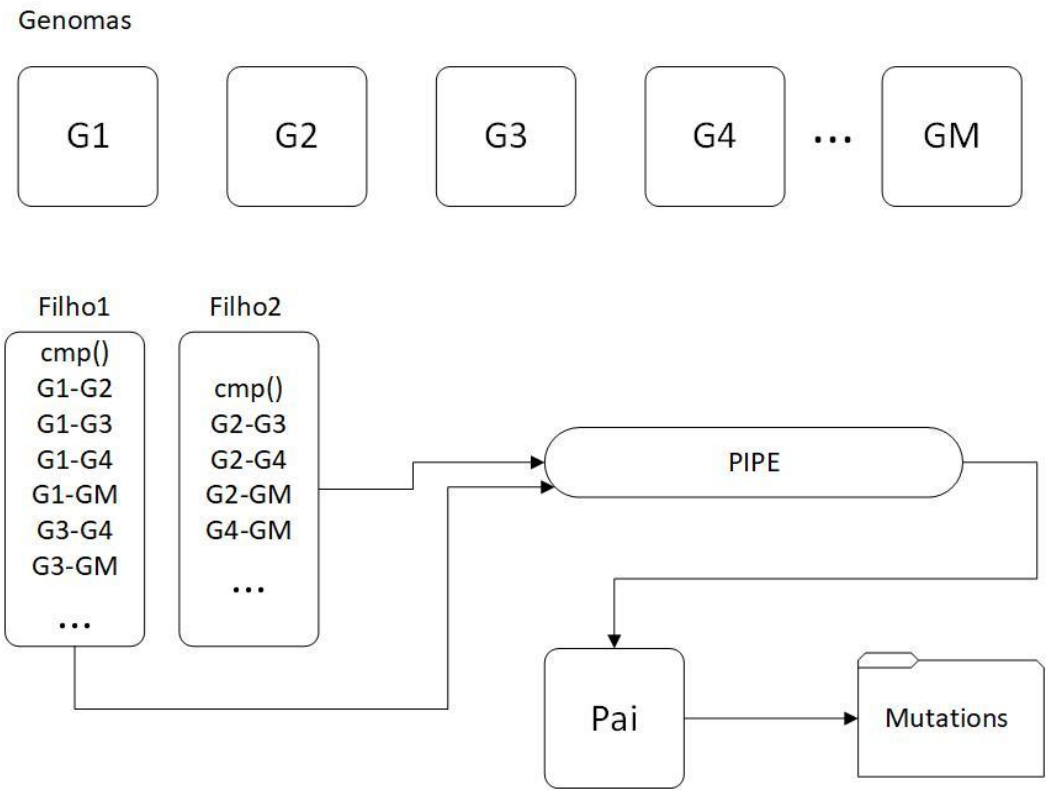


Figura 2 - Pipeline deteção mutações, requisito D

E. (25%) Esta etapa implica que o programa suporte a comunicação entre processos com recurso a *Unix Domin Sokects*[3]. Cada filho deve estabelecer conexão com o server (pai). O pai deve atender as conexões e armazenar as mutações detetadas em ficheiro.

3. Notas

Mantendo o mesmo contexto de deteção de mutações em genomas Sars-CoV-2, um segundo enunciado será oportunamente disponibilizado, desta vez com vista à avaliação dos conhecimentos adquiridos em sincronização de tarefas.

Este trabalho será realizado individualmente ou em grupos de dois alunos. O trabalho (README e código fonte) tem de ser submetido até à data indicada no sistema de elearning (trabalhos) e será apresentado e defendido de "viva voz" em data a designar pelo docente.

4. Bibliografia

[1] *Advanced Programming in the UNIX ® Environment* - Signal - 10.14 sigaction Function

[2] *Advanced Programming in the UNIX ® Environment* - Advanced I/O - 14.7 readn and writen Functions

[3] *Advanced Programming in the UNIX ® Environment* - Advanced IPC - 17.2 UNIX Domain Sockets

[4] *NCBI SARS-CoV-2 Resources* - https://www.ncbi.nlm.nih.gov/sars-cov-2/