

# Sistemas Operacionais

Busca Paralela em Arquivos com Multithreading

Abrahão Picanço e Lucas Gabriel

# Introdução



## O que é busca paralela em arquivos?

- Técnica para procurar dados em múltiplos arquivos simultaneamente.
- Usa multithreading para dividir o trabalho entre várias threads.

## Por que é importante?

- Acelera o processamento em grandes volumes de dados.
- Aproveita arquiteturas multicore modernas.

# O que é Multithreading?



## Definição

Multithreading é a capacidade de um programa executar múltiplas threads (linhas de execução) simultaneamente, cada uma realizando uma tarefa específica.

## Vantagens:

- Melhor utilização de CPUs multicore.
- Redução do tempo de execução para tarefas paralelizáveis.

## Aplicação na busca em arquivos:

 Cada thread processa um arquivo ou uma parte de um arquivo.

## Como Funciona a Busca Paralela?



#### 1. Divisão da tarefa:

- Lista de arquivos é dividida entre threads.
- Cada thread busca uma palavra-chave ou padrão em seu subconjunto.

## 2. Execução concorrente:

Threads leem e processam arquivos simultaneamente.

## 3. Agregação de resultados:

 Resultados de cada thread são combinados para formar a saída final.

## Vantagens da Busca Paralela



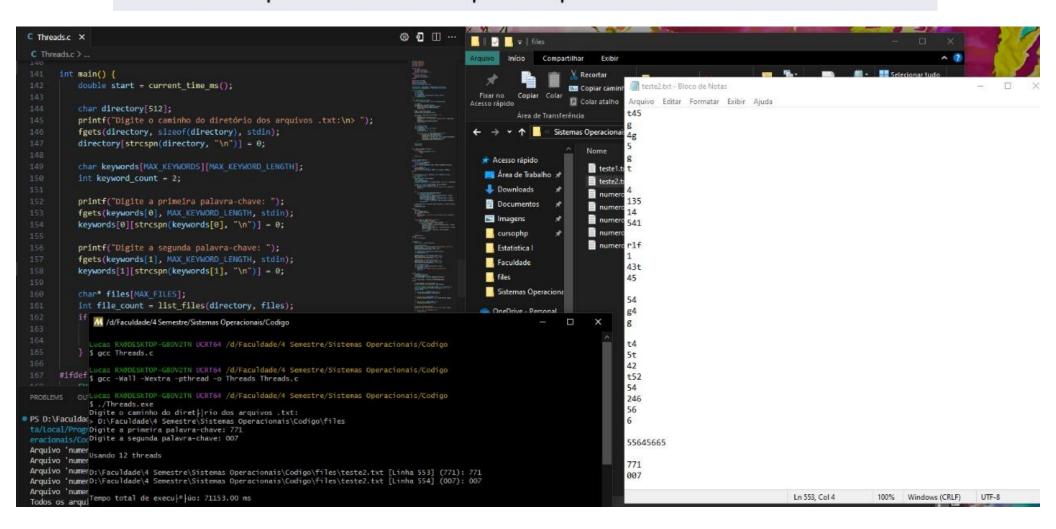
- Eficiência: Reduz significativamente o tempo de busca em grandes conjuntos de dados.
- Escalabilidade: Pode ser aplicada a milhares de arquivos ou grandes diretórios.
- Aproveitamento de hardware: Utiliza múltiplos núcleos de CPU.
- **Flexibilidade:** Pode ser adaptada para diferentes tipos de busca (ex.: regex, case-sensitive).

## **Exemplo Prático**



## Cenário

Buscar uma palavra em múltiplos arquivos de texto usando threads.



## **Exemplo Prático**



#### Cenário

Buscar uma palavra em múltiplos arquivos de texto usando threads.

```
⊕ • □ □ …
 EXPLORER
                          C Threads.c X
V OPEN EDITORS
                           C Threads.c > ...
  X C Threads.c
                                 int main() {

∨ CODIGO

                                      double start = current time ms();
 > .vscode
 ∨ files
                                      char directory[512];
                                      printf("Digite o caminho do diretório dos arquivos .txt:\n> ");

    ■ numeros 1k.txt

                                      fgets(directory, sizeof(directory), stdin);

■ numeros 5k.txt

                                      directory[strcspn(directory, "\n")] = 0;
  ■ numeros 10k.txt

■ numeros_50k.txt

                                      char keywords [MAX KEYWORDS] [MAX KEYWORD LENGTH];
  ■ numeros_100k.txt
                                      int keyword count = 2;

    teste1.txt

                                      printf("Digite a primeira palavra-chave: ");

    teste2.txt

                                      fgets(keywords[0], MAX KEYWORD LENGTH, stdin);
 > output
                                      keywords[0][strcspn(keywords[0], "\n")] = 0;

    a.exe

 ■ busca.exe
                                      printf("Digite a segunda palavra-chave: ");
 🕏 gerador nv
 🌈 📈 /d/Faculdade/4 Semestre/Sistemas Operacionais/Codigo
   Lucas RX@DESKTOP-G80V2TN UCRT64 /d/Faculdade/4 Semestre/Sistemas Operacionais/Codigo
  Digite o caminho do diret rio dos arquivos .txt:
  > D:\Faculdade\4 Semestre\Sistemas Operacionais\Codigo\files
  Digite a primeira palavra-chave: lucas
  Digite a segunda palavra-chave: lucas
  Usando 12 threads
  D:\Faculdade\4 Semestre\Sistemas Operacionais\Codigo\files\numeros_1k.txt [Linha 45] (lucas): lucas
  D:\Faculdade\4 Semestre\Sistemas Operacionais\Codigo\files\numeros_1k.txt [Linha 45] (lucas): lucas
  D:\Faculdade\4 Semestre\Sistemas Operacionais\Codigo\files\numeros_10k.txt [Linha 4489] (lucas): lucas
  D:\Faculdade\4 Semestre\Sistemas Operacionais\Codigo\files\numeros_10k.txt [Linha 4489] (lucas): lucas
  D:\Faculdade\4 Semestre\Sistemas Operacionais\Codigo\files\numeros_5k.txt [Linha 1194] (lucas): lucas
  D:\Faculdade\4 Semestre\Sistemas Operacionais\Codigo\files\numeros_5k.txt [Linha 1194] (lucas): lucas
  D:\Faculdade\4 Semestre\Sistemas Operacionais\Codigo\files\numeros_50k.txt [Linha 49824] (lucas): lucas
  D:\Faculdade\4 Semestre\Sistemas Operacionais\Codigo\files\numeros_50k.txt [Linha 49824] (lucas): lucas
  D:\Faculdade\4 Semestre\Sistemas Operacionais\Codigo\files\numeros_100k.txt [Linha 93218] (lucas): lucas
  D:\Faculdade\4 Semestre\Sistemas Operacionais\Codigo\files\numeros_100k.txt [Linha 93218] (lucas): lucas
  Tempo total de execulo úo: 16615.00 ms
```

## **Desafios e Cuidados**



#### Concorrência:

- Gerenciamento de acesso a recursos compartilhados (ex.: lista de resultados).
- Uso de mecanismos como Lock para evitar condições de corrida.

#### Erros de I/O:

Arquivos corrompidos ou inacessíveis podem causar falhas.

#### Overhead:

 Criar e gerenciar threads tem um custo computacional. Muitas threads podem saturar o sistema.

# **Aplicações Reais**



- Logs de servidores: Busca rápida por erros ou eventos específicos em logs.
- Motores de busca: Indexação paralela de documentos.
- Análise de dados: Processamento de grandes conjuntos de dados textuais.
- Ferramentas de desenvolvimento: Busca em código-fonte (ex.: IDEs).

## Conclusão



- A busca paralela com multithreading é uma técnica poderosa para acelerar o processamento de arquivos.
- Combina eficiência, escalabilidade e aproveitamento de hardware moderno.
- Requer cuidado com concorrência e gerenciamento de recursos.
- Aplicável em diversas áreas, desde análise de dados até ferramentas de software.