Projeto ‘grep paralelo’

Objetivo: encontrar quais arquivos contém um padrão (expressão regular), com a opção de ignorar diferença entre maiúsculas e minúsculas.

O fonte foi feito para executar a pesquisa de forma sequencial e posteriormente de forma paralela, a fim de comparar os tempos de execução.

Arquivos pesquisados:

1..989 arquivos texto com texto extraído das imprensas oficiais do município de Jundiaí, totalizando 361MB de texto.

Resultado da execução:

Pesquisando expressão 'Concurso.{0,50}Tempor' nos arquivos em

C:\x\txt

Número total de arquivos: 1989

Pré-processamento: 1323ms

Serial: 1825ms

Número de arquivos encontrados (serial): 3

Paralelo: 781ms

Número de arquivos encontrados (serial): 3

Padrão 'Concurso.{0,50}Tempor' encontrado nos seguintes arquivos:C:\x\txt

C:\x\txt\Imprensa\_Oficial\_Jundiai\_2018-10-20\_4468.txt

C:\x\txt\Imprensa\_Oficial\_Jundiai\_2019-09-11\_4607.txt

C:\x\txt\Imprensa\_Oficial\_Jundiai\_2017-06-28\_4285.txt

Código fonte:

using System;

using System.Text.RegularExpressions;

using System.Diagnostics;

using System.IO;

using System.Collections.Generic;

using System.Threading.Tasks;

using System.Threading;

// exerc\_grep caminho padrao

// - pesquisa pelo padrão de expressão regular 'padrao' em todos os arquivos da pasta 'caminho'

namespace exerc\_grep {

class Program {

static public Regex r;

static void Main(string[] args) {

Stopwatch sw; // cronômetro para tempo de execução

Mutex mutex = new Mutex(); // mutex para adicionar item à lista

r = new Regex(args[1], RegexOptions.IgnoreCase); // cria (compila) expressão regular

string[] fileEntries = Directory.GetFiles(args[0]); // pega lista de arquivos do diretório

Console.WriteLine("Pesquisando expressão '" + args[1] + "' nos arquivos em\n" + args[0]);

Console.WriteLine("Número total de arquivos: " + fileEntries.Length);

// Primeiro processamento - faz leitura de todos os arquivos apenas para carregar arquivos em caches

// -- como na primeira leitura os arquivos podem caber todos em cache na memória, lê todos os arquivos

// -- de modo que a medição de tempo serial e paralelo ocorram nas mesmas condições

sw = Stopwatch.StartNew();

foreach (var f in fileEntries) {

string texto = File.ReadAllText(f, System.Text.Encoding.ASCII);

}

sw.Stop();

Console.WriteLine("Pré-processamento: {0}ms", sw.ElapsedMilliseconds);

// realiza pesquisa no conteúdo dos arquivos de forma sequencial

sw = Stopwatch.StartNew();

int s\_NumArqs = 0;

var s\_achados = new List<string>(); // lista para armazenar nomes dos arquivos encontrados

foreach (var f in fileEntries) {

string texto = File.ReadAllText(f, System.Text.Encoding.ASCII); // lê arquivo usando codificação ASCII

if (r.IsMatch(texto)) { // caso encontre o padrão da expressão regular no arquivo,

s\_NumArqs += 1; // incrementa contador

s\_achados.Add(f); // adiciona à lista de arquivos encontrados

}

}

sw.Stop();

Console.WriteLine("Serial: {0}ms", sw.ElapsedMilliseconds);

Console.WriteLine("Número de arquivos encontrados (serial): {0}", s\_NumArqs);

// realiza mesma pesquisa de forma paralela

sw = Stopwatch.StartNew();

int p\_NumArqs = 0;

var p\_achados = new List<string>();

var opt = new ParallelOptions() { // variável com opções de paralelismo

MaxDegreeOfParallelism = 4 // Opção para número máximo de threads

};

Parallel.ForEach(fileEntries, opt, f => {

string texto = File.ReadAllText(f, System.Text.Encoding.ASCII); // lê arquivo usando codificação ASCII

if (r.IsMatch(texto)) { // caso encontre o padrão da expressão regular no arquivo,

Interlocked.Increment(ref p\_NumArqs); // Operação de incremento de forma atômica

mutex.WaitOne(); // mutex para executar método na lista p\_achados

p\_achados.Add(f);

mutex.ReleaseMutex();

}

});

sw.Stop();

Console.WriteLine("Paralelo: {0}ms", sw.ElapsedMilliseconds);

Console.WriteLine("Número de arquivos encontrados (serial): {0}", s\_NumArqs);

// Apresenta arquivos onde encontrou o padrão

Console.WriteLine("Padrão '" + args[1] + "' encontrado nos seguintes arquivos:" + args[0]);

foreach (var t in p\_achados)

Console.WriteLine(t);

}

}

}