加速下载体验：C#多线程分块下载文件与实时进度展示

作者：微信公众号：【架构师老卢】 12-1 20:49 128

概述：该C#示例演示了如何使用多线程分块下载文件并显示下载进度。程序通过确定文件大小，创建多个线程，分配下载范围，同时下载文件块，最后合并文件。通过简单的控制台应用，用户可以清晰地看到下载进度。此方法提高了下载效率，更好地利用了网络带宽。

多线程分块下载文件的原理是将文件分成多个块，每个线程负责下载一个块的数据，最后将所有块合并成完整的文件。这样可以提高下载速度，并充分利用网络带宽。

方法与步骤

1. **确定下载文件的大小：** 在下载之前，需要获取要下载文件的大小，以便将其分成适当的块。
2. **创建多个线程：** 创建多个线程来同时下载不同的文件块。可以使用`Thread`类或`Task`类。
3. **分配每个线程的下载范围：** 将文件大小平均分配给每个线程，确保每个线程下载不同的文件块。
4. **下载文件块：** 每个线程根据分配的范围下载文件块，然后将其保存到本地。
5. **等待所有线程完成：** 使用线程同步机制，确保所有线程都完成下载任务。
6. **合并文件块：** 将下载的文件块按照顺序合并成完整的文件。
7. **显示下载进度：** 可以使用委托或事件来更新下载进度，确保用户能够看到下载的进展情况。

完整实例

以下是一个简单的C#控制台应用程序，用于演示多线程分块下载文件并显示进度。

```
using System;
using System.IO;
using System.Net;
using System.Threading;

class Program
{
    static int numThreads = 4; // 可以根据需要设置线程数
    static long fileSize;
    static long blockSize;
    static long downloadedSize = 0;

    static void Main()
    {
        string fileUrl = "https://example.com/largefile.zip";
        string savePath = "downloadedFile.zip";

        // 获取文件大小
        fileSize = GetFileSize(fileUrl);

        // 计算每个线程下载的块大小
        blockSize = fileSize / numThreads;

        // 创建线程数组
        Thread[] threads = new Thread[numThreads];

        // 下载文件并显示进度
        for (int i = 0; i < numThreads; i++)
        {
            int threadNumber = i;
            threads[i] = new Thread(() => DownloadFilePart(fileUrl, savePath, threadNumber));
            threads[i].Start();
        }

        // 等待所有线程完成
        foreach (var thread in threads)
        {
            thread.Join();
        }

        Console.WriteLine("下载完成！");
    }

    static void DownloadFilePart(string fileUrl, string savePath, int threadNumber)
    {
        long startByte = threadNumber * blockSize;
        long endByte = (threadNumber == numThreads - 1) ? fileSize - 1 : startByte + blockSize - 1;

        WebClient client = new WebClient();
        Stream stream = client.OpenRead(fileUrl);

        // 设置读取的起始位置
        stream.Seek(startByte, SeekOrigin.Begin);

        // 创建文件流用于保存下载的块
        using (FileStream fs = new FileStream(savePath, FileMode.OpenOrCreate, FileAccess.Write, FileShare.Write))
        {
            byte[] buffer = new byte[1024];
            int bytesRead;

            while ((bytesRead = stream.Read(buffer, 0, buffer.Length)) > 0)
            {
                fs.Write(buffer, 0, bytesRead);
                Interlocked.Add(ref downloadedSize, bytesRead);
                DisplayProgress();
            }
        }

        stream.Close();
    }

    static void DisplayProgress()
    {
        double progress = (double)downloadedSize / fileSize * 100;
        Console.WriteLine($"已下载：{progress:F2}%");
    }

    static long GetFileSize(string fileUrl)
    {
        WebRequest request = WebRequest.Create(fileUrl);
        request.Method = "HEAD";

        using (WebResponse response = request.GetResponse())
        {
            long contentLength;
            if (long.TryParse(response.Headers.Get("Content-Length"), out contentLength))
            {
                return contentLength;
            }
            else
            {
                throw new InvalidOperationException("无法获取文件大小。");
            }
        }
    }
}
```

请注意，此示例使用了`WebClient`和`WebRequest`类来下载文件。在实际应用中，可能需要处理更多的异常情况，并根据需要调整代码。此外，为了简化示例，没有包含对HTTPS、重试机制等的处理。在生产环境中，这些方面需要更多的注意。

相关代码下载地址