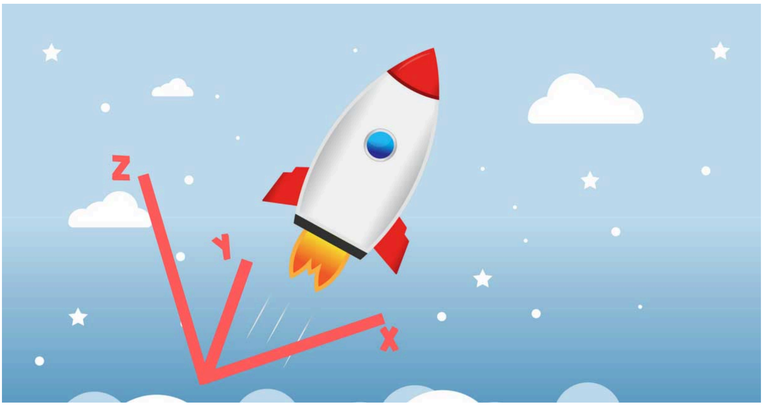
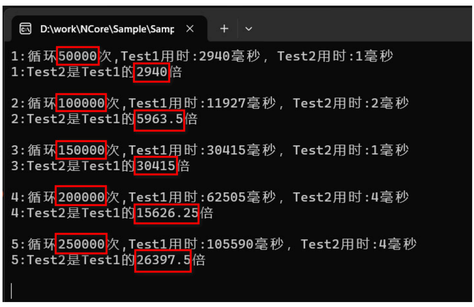
改一个对象类型，for循环耗时从3000毫秒下降到1毫秒，性能优化很重要！

作者：微信公众号：【架构师老卢】 12-12 7:43 174



概述：在C#中，字符串连接有两种实现方法：使用`+`运算符和使用`StringBuilder`。前者在每次连接时都会创建新的字符串对象，效率较低。后者通过内部管理字符数组，避免了频繁的内存分配和垃圾回收，因此性能更高。在处理大量字符串连接时，使用`StringBuilder`可以显著提高性能。这两种方法在功能上等价，但性能差异可达10倍或更多。 。

先上效果:



最近在和网友聊天时他问道：他做了一个生成代码的小工具，生成一个文件很快，但生成一个项目时就会很慢，找不到原因，让我帮分析一下是哪里的问题。能过性能分析工具和查看相关代码，发现他大量使用了字符串拼接，问题就出在这里了，下面来分析一下。

在C#中，字符串拼接时使用 `string` 和 `StringBuilder` 会导致性能差异的主要原因是，`string` 类型是不可变的，每次拼接都会创建一个新的字符串对象，而 `StringBuilder` 是可变的，可以在原始对象上进行操作，避免了创建新对象的开销。

下面分别演示使用 `string` 和 `StringBuilder` 进行字符串拼接的性能差异，并提供详细的实例源代码。

使用string 和 StringBuilder 进行字符串拼接：

```
public static class Program
{
    static void Main(string[] args)
    {
        //循环50000次
        int start = 50000;

        //测试5次每以50000的数量增加
        for (int i = 0; i < 5; i++)
        {
            //循环次数
            int end = start + (start * i);

            //测量执行时间(单位为毫秒)
            var executionTimer = GetExecutionTimer(() =>
            {
                //执行测试
                Test1(end);
            });

            //测量执行时间(单位为毫秒)
            var executionTimer2 = GetExecutionTimer(() =>
            {
                //执行测试
                Test2(end);
            });

            Console.WriteLine($"{(i + 1)}:循环{end}次,Test1用时:{executionTimer}毫秒, Test2用时:{executionTimer2}毫秒");
            Console.WriteLine($"{(i + 1)}:Test2是Test1的{((double)executionTimer / executionTimer2)}倍");
            Console.WriteLine();

        }
        Console.ReadKey();
    }

    /// <summary>
    /// 测试方法1
    /// </summary>
    static void Test1(int end)
    {
        string result = "";
        for (int i = 0; i < end; i++)
        {
            result += i.ToString();
        }
    }

    /// <summary>
    /// 测试方法2
    /// </summary>
    static void Test2(int end)
    {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < end; i++)
        {
            sb.Append(i);
        }
        string result = sb.ToString();
    }


    /// <summary>
    /// 返回一个委托执行时间(通用)
    /// </summary>
    /// <param name="action">要执行的代码块</param>
    /// <returns>代码块的执行时间（毫秒）</returns>
    static long GetExecutionTimer(this Action action)
    {
        // 获取当前时间戳
        var stopwatch = new Stopwatch();
        stopwatch.Start();

        // 执行传入的代码块
        action();

        // 停止计时
        stopwatch.Stop();

        // 返回执行时间
        return stopwatch.ElapsedMilliseconds;
    }
}
```

上述两个示例中，使用 `string` 拼接字符串时，每次循环都会创建一个新的字符串对象，而使用 `StringBuilder` 则会在原始对象上进行追加，避免了创建多个对象。在迭代次数较多时，`StringBuilder` 的性能明显优于直接使用 `string`。

源代码获取：公众号回复消息【code：63109】

相关代码下载地址

重要提示！：取消关注公众号后将无法再启用回复功能，不支持解封！

第一步：微信扫码关键公众号“架构师老卢”

第二步：在公众号聊天框发送 code：63109 ，如：code：63109 获取下载地址

第三步：恭喜你，快去下载你想要的资源吧