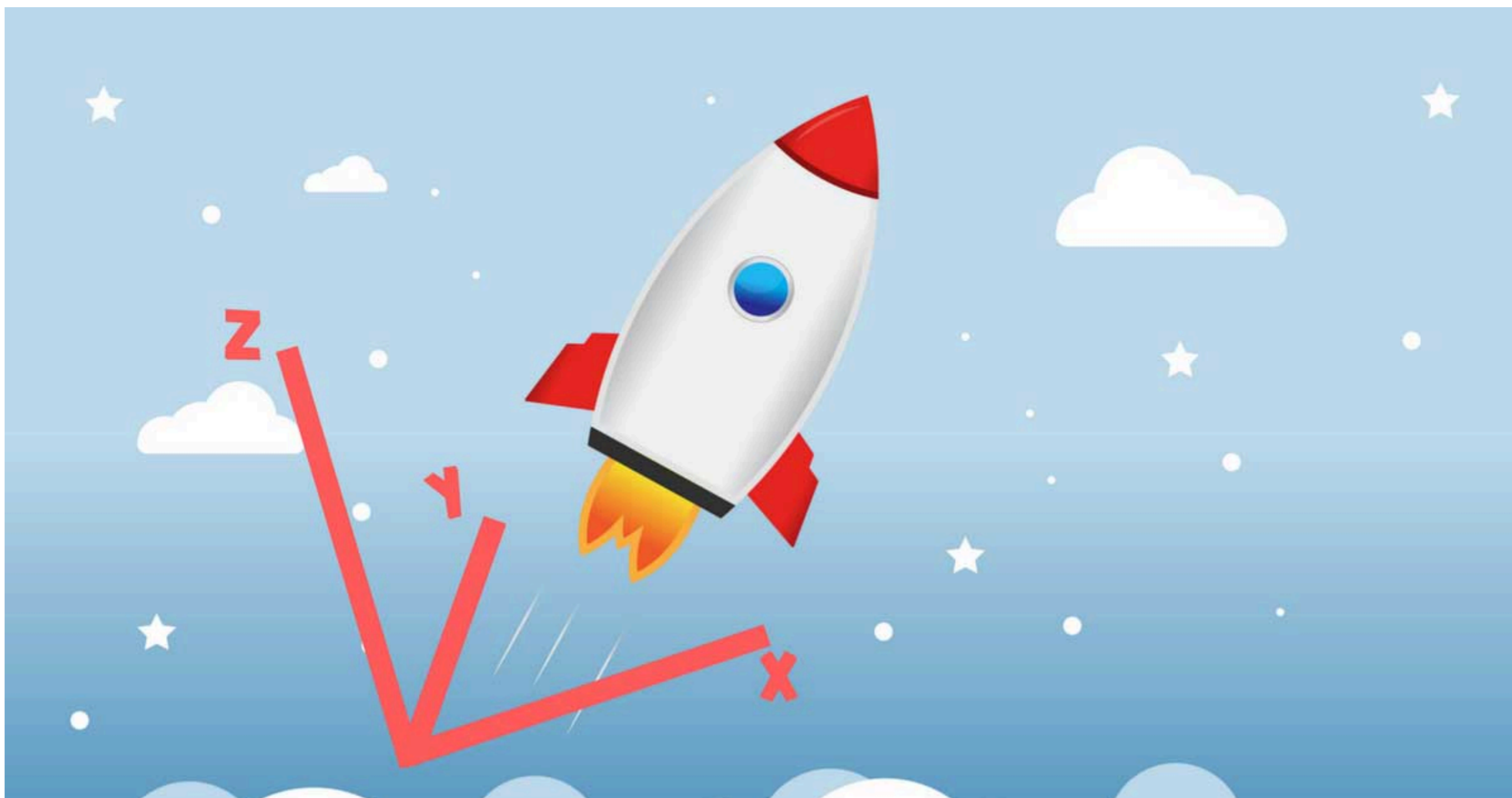


改一个对象类型，for循环耗时从3000毫秒下降到1毫秒，性能优化很重要！

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，避免了频繁的内部分配和垃圾回收，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。

```
D:\work\NCore\Sample\Samp... x + v
1: 循环50000次, Test1用时: 2940毫秒, Test2用时: 1毫秒
1: Test2是Test1的2940倍
2: 循环100000次, Test1用时: 11927毫秒, Test2用时: 2毫秒
2: Test2是Test1的5963.5倍
先上效果: 3: 循环150000次, Test1用时: 30415毫秒, Test2用时: 1毫秒
3: Test2是Test1的30415倍
4: 循环200000次, Test1用时: 62505毫秒, Test2用时: 4毫秒
4: Test2是Test1的15626.25倍
5: 循环250000次, Test1用时: 105590毫秒, Test2用时: 4毫秒
5: Test2是Test1的26397.5倍
```

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：

```
1 public static class Program
2 {
3     static void Main(string[] args)
4     {
5         //循环50000次
6         int start = 50000;
7
8         //测试5次每以50000的数量增加
9         for (int i = 0; i < 5; i++)
10        {
11            //循环次数
12            int end = start + (start * i);
13
14            //测量执行时间(单位为毫秒)
15            var executionTimer = GetExecutionTimer(() =>
16            {
17                //执行测试
18                Test1(end);
19            });
20
21            //测量执行时间(单位为毫秒)
22            var executionTimer2 = GetExecutionTimer(() =>
23            {
24                //执行测试
25                Test2(end);
26            });
27
28            Console.WriteLine($"{i + 1}): 循环{end}次, Test1用时:{executionTimer}毫秒, Test2用时:{executionTimer2}毫秒");
29            Console.WriteLine($"{i + 1}): Test2是Test1的{((double)executionTimer / executionTimer2)}倍");
30            Console.WriteLine();
31        }
32        Console.ReadKey();
33    }
34
35    /// <summary>
36    /// 测试方法1
37    /// </summary>
38    static void Test1(int end)
39    {
40        string result = "";
41        for (int i = 0; i < end; i++)
42        {
43            result += i.ToString();
44        }
45    }
46
47    /// <summary>
48    /// 测试方法2
49    /// </summary>
50    static void Test2(int end)
51    {
52        StringBuilder sb = new StringBuilder();
53        for (int i = 0; i < end; i++)
54        {
55            sb.Append(i);
56        }
57        string result = sb.ToString();
58    }
59
60    /// <summary>
61    /// 返回一个委托执行时间(通用)
62    /// </summary>
63    /// <param name="action">要执行的代码块</param>
64    /// <returns>代码块的执行时间(毫秒)</returns>
65    static long GetExecutionTimer(this Action action)
66    {
67        // 获取当前时间戳
68        var stopwatch = new Stopwatch();
69        stopwatch.Start();
70
71        // 执行传入的代码块
72        action();
73
74        // 停止计时
75        stopwatch.Stop();
76
77        // 返回执行时间
78        return stopwatch.ElapsedMilliseconds;
79    }
80
81 }
82 }
```

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: 微信扫码关注公众号“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