公告

网名：webabcd
本名：王磊
位置：中国 北京
荣誉：微软最有价值专家（MVP）

昵称：webabcd
园龄：9年4个月
荣誉：推荐博客
粉丝：1940
关注：0
+加关注

搜索

找找看

常用链接

我的随笔
我的评论
我的参与

精进不休 .NET 4.0 (5) - C# 4.0 新特性之并行运算(Parallel)
Posted on 2010-06-03 08:37 webabcd 阅读(15390) 评论(19) 编辑 收藏

[索引页]
[源码下载]

精进不休 .NET 4.0 (5) - C# 4.0 新特性之并行运算(Parallel)

作者：webabcd

介绍
C# 4.0 的新特性之并行运算

- Parallel.For - for 循环的并行运算
- Parallel.ForEach - foreach 循环的并行运算
- Parallel.Invoke - 并行调用多个任务
- Task - 任务，基于线程池。其使我们对并行编程变得更简单，且不用关心底层是怎么实现的
- PLINQ - 用于对内存中的数据做并行运算，也就是说其只支持 LINQ to Object 的并行运算

示例
1、Parallel.For 的 Demo
Parallel/ParallelFor.aspx.cs

最新评论
我的标签
更多链接

代码

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace CSharp.Parallel
{
    public partial class ParallelFor : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Normal();
            ParallelForDemo();
        }

        private void Normal()
        {
            DateTime dt = DateTime.Now;

            for (int i = 0; i < 20; i++)
            {
                GetData(i);
            }

            Response.Write((DateTime.Now - dt).TotalMilliseconds.ToString());
            Response.Write("<br />");
            Response.Write("<br />");
        }

        private void ParallelForDemo()
        {
            DateTime dt = DateTime.Now;

            // System.Threading.Tasks.Parallel.For - for 循环的并行运算
```

YYControls - 媛媛控件(17)
YYGames - Silverlight游戏(8)

随笔档案(486)

2015年7月 (6)
2015年6月 (5)
2015年5月 (5)
2015年4月 (5)
2015年3月 (5)
2015年1月 (5)
2014年8月 (6)
2014年7月 (5)
2014年6月 (6)
2014年5月 (5)
2014年2月 (1)
2014年1月 (7)
2013年12月 (9)
2013年11月 (6)
2013年10月 (6)
2013年9月 (7)
2013年8月 (9)
2013年7月 (9)
2013年6月 (7)
2013年5月 (8)
2013年4月 (7)
2013年3月 (8)
2013年1月 (7)
2012年10月 (4)
2012年9月 (8)
2012年8月 (9)

```
            System.Threading.Tasks.Parallel.For(0, 20, (i) => { GetData(i); });

            Response.Write((DateTime.Now - dt).TotalMilliseconds.ToString());
            Response.Write("<br />");
        }

        private int GetData(int i)
        {
            System.Threading.Thread.Sleep(100);
            Response.Write(i.ToString());
            Response.Write("<br />");
            return i;
        }
    }
}

/*
运行结果：
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

```
19
2000.0514

0
13
1
19
7
12
18
6
2
8
10
14
4
16
5
3
15
17
9
11
300.0077
*/
```

2、Parallel.ForEach 的 Demo
Parallel/ParallelForEach.aspx.cs

代码

```
using System;
using System.Collections.Generic;
```

2008年5月 (2)
2008年4月 (6)
2008年2月 (3)
2008年1月 (3)
2007年12月 (3)
2007年11月 (3)
2007年10月 (15)
2007年9月 (3)
2007年8月 (5)
2007年7月 (5)
2007年6月 (10)
2007年5月 (19)
2007年4月 (9)
2007年3月 (8)
2007年2月 (20)
2007年1月 (18)
2006年12月 (7)

相册(27)

DesignPattern设计模式(24)
YYControls媛媛控件库(3)

收藏夹

APP HUB
asp.net
Channel 9
CodePlex
CodeProject
Microsoft Research

```
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace CSharp.Parallel
{
    public partial class ParallelForEach : System.Web.UI.Page
    {
        private List<int> _data = new List<int>();

        protected void Page_Load(object sender, EventArgs e)
        {
            InitData();

            Normal();
            ParallelForEachDemo();
        }

        private void InitData()
        {
            _data.Clear();
            for (int i = 0; i < 20; i++)
            {
                _data.Add(i);
            }
        }

        private void Normal()
        {
            DateTime dt = DateTime.Now;

            for (int i = 0; i < 20; i++)
            {
                GetData(i);
            }
```

Microsoft Silverlight
MS Forums
MS Forums China
MSDN Blogs
MSDN Code Gallery
msdn en
msdn zh
Scott Guthrie

积分与排名

积分 - 2710958

排名 - 10

最新评论

1. Re:重新想象 Windows 8 Store Apps (12) - 控件之 GridView 特性: 拖动项, 项尺寸可变, 分组显示

@bloodypanda嗯，也许部分功能不支持，用win10测试看看吧...

--webabcd

2. Re:重新想象 Windows 8 Store Apps (12) - 控件之 GridView 特性: 拖动项, 项尺寸可变, 分组显示

请问windows phone8.1,实现上述gridview拖放代码，没有报错，但是运行结果不对，仍然不能进行拖放。

--bloodypanda

3. Re:重新想象 Windows 8 Store Apps (30) - 信息: 获取包信息, 系统信息, 硬件信息, PnP信息, 常用设备信息

@hippieZhou参考 中的 “通过 C++

```
            Response.Write((DateTime.Now - dt).TotalMilliseconds.ToString());
            Response.Write("<br />");
            Response.Write("<br />");
        }

        private void ParallelForEachDemo()
        {
            DateTime dt = DateTime.Now;

            // System.Threading.Tasks.Parallel.ForEach - foreach 循环的并行运算
            System.Threading.Tasks.Parallel.ForEach(_data, (index) => { GetData(index); });

            Response.Write((DateTime.Now - dt).TotalMilliseconds.ToString());
            Response.Write("<br />");
        }

        private int GetData(int i)
        {
            System.Threading.Thread.Sleep(100);
            Response.Write(i.ToString());
            Response.Write("<br />");
            return i;
        }
    }
}

/*
运行结果：
0
1
2
3
4
5
6
7
8
```

和 D3D 获取屏幕分辨率”...

--webabcd

4. Re:重新想象 Windows 8 Store Apps (30) - 信息: 获取包信息, 系统信息, 硬件信息, PnP信息, 常用设备信息

请问如何获取当前运行设备的分辨率？？？？

--hippieZhou

5. Re:系出名门 Android 系列文章索引

@顾晓北:)是我的入门笔记...

--webabcd

6. Re:重新想象 Windows 8 Store Apps (30) - 信息: 获取包信息, 系统信息, 硬件信息, PnP信息, 常用设备信息

@hplucky这个我不知道...

--webabcd

7. Re:系出名门 Android 系列文章索引

Android入门经典？哈哈

--顾晓北

8. Re:重新想象 Windows 8 Store Apps (30) - 信息: 获取包信息, 系统信息, 硬件信息, PnP信息, 常用设备信息

补充上面：是在universal app里获取removable device的信息，比如是usb还是手机，手机系统等等信息。

--hplucky

9. Re:重新想象 Windows 8 Store Apps (30) - 信息: 获取包信息, 系统信息, 硬件信息, PnP信息, 常用设备信息

```
9
10
11
12
13
14
15
16
17
18
19
2000.0514

0
6
12
18
1
2
7
13
19
4
3
8
14
9
5
15
10
16
11
17
600.0154
*/
```

请问如何获取设备信息，比如是usb还是手机？手机系统是什么等等这些信息？ 谢谢。

--hplucky

10. Re:重新想象 Windows 8.1 Store Apps (88) - 通信的新特性: 新的 HttpClient

@安静的豆子结合HttpBaseProtocolFilter来实现，它有一个AllowAutoRedirect属性...

--webabcd

11. Re:重新想象 Windows 8.1 Store Apps (88) - 通信的新特性: 新的 HttpClient

您好，请教一个问题。在windows.web.http.httpclient 里有没有什么办法实现阻止302跳转实现（在Windowsphone里遇到的，一直没搞清），希望得到解答，谢谢！

--安静的豆子

12. Re:不可或缺 Windows Native (1) - C 语言: hello c

@深蓝医生:)多谢支持...

--webabcd

13. Re:不可或缺 Windows Native 系列文章索引

@陶子@conan_lin:)多谢支持...

--webabcd

14. Re:重新想象 Windows 8.1 Store Apps (88) - 通信的新特性: 新的 HttpClient

@sun_yang启webview调用mailto（不能发附件）通过"Share Contract"的方式，调用本地mail客户端发找个第三方的类库自己写socket实现一个自己的smtp...

3、Parallel.Invoke 的 Demo

Parallel/ParallelInvoke.aspx.cs

代码

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

using System.Threading;

namespace CSharp.Parallel
{
    public partial class ParallelInvoke : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            var tasks = new Action[] { () => Task1(), () => Task2(), () => Task3() };

            // System.Threading.Tasks.Parallel.Invoke - 并行调用多个任务
            System.Threading.Tasks.Parallel.Invoke(tasks);
        }

        private void Task1()
        {
            Thread.Sleep(3000);
            Response.Write("Task1 - " + "ThreadId:" + Thread.CurrentThread.ManagedThreadId.ToString()
            Response.Write("<br />");
        }

        private void Task2()
        {
```

--webabcd

15. Re:不可或缺 Windows Native (1) - C 语言: hello c

Mark

--深蓝医生

```
            System.Threading.Thread.Sleep(3000);
            Response.Write("Task2 - " + "ThreadId:" + Thread.CurrentThread.ManagedThreadId.ToString()
            Response.Write("<br />");
        }

        private void Task3()
        {
            System.Threading.Thread.Sleep(3000);
            Response.Write("Task3 - " + "ThreadId:" + Thread.CurrentThread.ManagedThreadId.ToString()
            Response.Write("<br />");
        }
    }
}

/*
运行结果：
Task2 - ThreadId:26 - 09:11:58
Task1 - ThreadId:25 - 09:11:58
Task3 - ThreadId:24 - 09:11:58
*/
```

4、Task 的 Demo

Parallel/ParallelTask.aspx.cs

```
代码
/*
Task - 任务，基于线程池。其使我们对并行编程变得更简单，且不用关心底层是怎么实现的
*/

using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

using System.Threading;
using System.Threading.Tasks;

namespace CSharp.Parallel
{
    public partial class ParallelTask : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            /*
             * CancellationTokenSource - 取消任务的操作需要用到的一个类
             *     Token - 一个 CancellationToken 类型的对象，用于通知取消指定的操作
             *     IsCancellationRequested - 是否收到了取消操作的请求
             *     Cancel() - 结束任务的执行
             * ParallelOptions - 并行运算选项
             *     CancellationToken - 设置一个 Token，用于取消任务时的相关操作
             *     MaxDegreeOfParallelism - 指定一个并行循环最多可以使用多少个线程
             */

            CancellationTokenSource cts = new CancellationTokenSource();
            ParallelOptions pOption = new ParallelOptions() { CancellationToken = cts.Token };
            pOption.MaxDegreeOfParallelism = 10;

            Response.Write("开始执行，3.5 秒后结束");
            Response.Write("<br />");

            /*
             * Task - 任务类
             *     Factory.StartNew() - 创建并开始一个或一批新任务
             *     ContinueWith() - 此任务完成后执行指定的另一个任务
             *     AsyncState - 此任务的上下文对象
             *     Wait() - 阻塞，直到任务完成
             */
```

```
Task task0 = Task.Factory.StartNew(() =>
{
    Thread.Sleep(3500);
    cts.Cancel();
    Response.Write("结束");
    Response.Write("<br />");

});

// 通过 System.Threading.Tasks.Parallel.Invoke 执行任务的时候，可以加入 ParallelOptions 参数，用
System.Threading.Tasks.Parallel.Invoke(pOption,
    () => Task1(pOption.CancellationToken),
    () => Task2(pOption.CancellationToken));


/*
 * 一个 Task 内可以包含多个 Task
Task tasks = new Task(() =>
{
    Task.Factory.StartNew(() => Method());
    Task.Factory.StartNew(() => Method2());
    Task.Factory.StartNew(() => Method3());
});
tasks.Start();
// 阻塞，直到整个任务完成
tasks.Wait();
*/


/*
 * 带返回值的 Task
Func<object, long> fun = delegate(object state)
{
    return 1.0;
};
Task<long> tsk = new Task<long>(fun, "state");
```

```
            tsk.Start();
            Response.Write(tsk.Result.ToString());
            */
        }

        private void Task1(CancellationToken token)
        {
            // 每隔 1 秒执行一次，直到此任务收到了取消的请求
            // 注意：虽然此处是其他线程要向主线程（UI线程）上输出信息，但因为使用了 Task ，所以不用做任何处理
            while (!token.IsCancellationRequested)
            {
                Response.Write("Task1 - " + "ThreadId: " + Thread.CurrentThread.ManagedThreadId.ToStri
                Response.Write("<br />");
                Thread.Sleep(1000);
            }

        }
        private void Task2(CancellationToken token)
        {
            while (!token.IsCancellationRequested)
            {
                Response.Write("Task2 - " + "ThreadId: " + Thread.CurrentThread.ManagedThreadId.ToStri
                Response.Write("<br />");
                Thread.Sleep(1000);
            }
        }
    }
}

/*
运行结果：
开始执行，3.5 秒后结束
Task2 - ThreadId: 6
Task1 - ThreadId: 48
Task1 - ThreadId: 48
Task2 - ThreadId: 6
Task2 - ThreadId: 6
```

```
Task1 - ThreadId: 48
Task2 - ThreadId: 6
Task1 - ThreadId: 48
结束
*/
```

5、PLINQ 的 Demo
Parallel/ParallelPLINQ.aspx.cs

代码

```
/*
PLINQ - 用于对内存中的数据做并行运算，也就是说其只支持 LINQ to Object 的并行运算
*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace CSharp.Parallel
{
    public partial class ParallelPLINQ : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            List<int> list = new List<int>();
            for (int i = 0; i < 100; i++)
            {
                list.Add(i);
            }
```

```
            // AsParallel() - 并行运算
            // AsSequential() - 串行运算
            // AsOrdered() - 保持数据的原有顺序（AsSequential()指的是串行运算；AsOrdered()指的是如果在并行运算的
然后排序，最后再把排序后的数据做输出）
            // AsUnordered() - 可以不必保持数据的原有顺序
            // WithDegreeOfParallelism() - 明确地指出需要使用多少个线程来完成工作
            // WithCancellation(new CancellationTokenSource().Token) - 指定一个 CancellationToken 类型的参

            ParallelQuery nums = from num in list.AsParallel<int>().AsOrdered<int>()
                                 where num % 10 == 0
                                 select num;

            foreach (var num in nums)
            {
                Response.Write(num.ToString());
                Response.Write("<br />");
            }

            // 聚合方法也可以做并行运算
            Response.Write(list.AsParallel().Average().ToString());
            Response.Write("<br />");

            // 自定义聚合方法做并行运算的 Demo（实现一个取集合的平均值的功能）
            double myAggregateResult = list.AsParallel().Aggregate(
                // 聚合变量的初始值
                0d,

                // 在每个数据分区上，计算此分区上的数据
                // 第一个参数：对应的数据分区的计算结果；第二个参数：对应的数据分区的每个数据项
                (value, item) =>
                {
                    double result = value + item;
                    return result;
                },

                // 根据每个数据分区上的计算结果，再次做计算
```

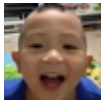
```
                // 第一个参数：全部数据的计算结果；第二个参数：每个数据分区上的计算结果
                (value, data) =>
                {
                    double result = value + data;
                    return result;
                },

                // 根据全部数据的计算结果再次计算，得到最终的聚合结果
                (result) => result / list.Count
            );

            Response.Write(myAggregateResult.ToString());
        }
    }
}

/*
运行结果：
0
10
20
30
40
50
60
70
80
90
49.5
49.5
*/
```

注：关于并行运算的实例可以参考
http://code.msdn.microsoft.com/ParExtSamples

OK
[源码下载]

分类: C#, LINQ

绿色通道： **好文要顶** **关注我** **收藏该文** **与我联系**



webabcd
关注 - 0
粉丝 - 1940
荣誉：推荐博客
+加关注

« 上一篇：精进不休 .NET 4.0 (4) - C# 4.0 新特性之命名参数和可选参数, 动态绑定(dynamic), 泛型协变和逆变, CountdownEvent, B
» 下一篇：精进不休 .NET 4.0 (6) - ADO.NET Data Services 1.5(WCF Data Services) 新特性

Feedback

#1楼 回复 引用
2010-06-03 09:55 by Silenus-G

学习了~非常棒的东东

#2楼 回复 引用
2010-06-03 11:34 by leeolevis

顶，支持

#3楼[楼主]　回复　引用
2010-06-03 12:58 by webabcd

@Silenus-G
@leeolevis
:)
多谢支持

#4楼　回复　引用
2010-06-03 13:41 by 君之蘭

学习

#5楼　回复　引用
2010-06-03 15:05 by woiwoqq

顶了，这个并行预算可以看作是在线程的基础上再封装一层吧

#6楼[楼主]　回复　引用
2010-06-03 17:39 by webabcd

@麦子|君子兰
:)
互相学习

#7楼[楼主]　回复　引用
2010-06-03 17:39 by webabcd

@woiwoqq
:)
多谢支持

嗯，是的，是在线程池的基础上再次封装的

#8楼　回复　引用
2010-06-10 11:11 by 方路

Threading，我这个八九年经验的老程序员还真没怎么用过线程。在与数据库打交道中，基本上用不到(或者说人没有第三只眼，跟本不知道多
步处理的东西都是用Timer实现。

#9楼[楼主]　回复　引用
2010-06-10 13:20 by webabcd

@方路
:)
确实，如果只是增删改查的话，比较少用到thread

#10楼　回复　引用
2010-06-18 00:05 by skyaspnet

源代码下载后提示压缩文件损坏，请楼主检查一下，谢谢！

#11楼[楼主]　回复　引用
2010-06-18 08:22 by webabcd

@skyaspnet
抱歉，确实有问题，已经修正了
多谢提醒

咳，skydrive不太好用，想保存在博客的空间里，容量又不够了

#12楼　回复　引用
2010-06-28 13:55 by 完成V

联系管理员，空间开大点呗。

#13楼[楼主]　回复　引用
2010-06-29 08:02 by webabcd

@№完成
:)
就不麻烦他们了，习惯之后skydrive还是很不错的

#14楼　回复　引用
2010-06-29 12:49 by frcsun

这些都是源代码啊。。。是哪里的？微软提供的示例还是LZ自己写的啊？
突然有个想法 解析下微软示例的源码，嗯~~~做个专题~~~

#15楼[楼主]　回复　引用
2010-06-29 16:42 by webabcd

@frcsun
自己写的

兄弟想法不错，很值得写一下

#16楼　回复　引用
2010-12-23 15:28 by Teddy

测试ParallelFor时，发现结果如下：

0
1
2
3
4
5
6
7
8
9
10

11
12
13
14
15
16
17
18
19
2015.6766

00
1
11
22
4
14
13
3
5
15
16
6
7
18
8
99
703.143

将GetData函数改为如下，测试正确：

```
private int GetData(int i)
{
System.Threading.Thread.Sleep(100);
Response.Write(i.ToString() + "<br />");
return i;
}
```

多谢楼主分享！

#17楼[楼主]　回复　引用
2010-12-24 11:39 by webabcd

@Teddy
:)
不谢，大家多交流

#18楼　回复　引用
2012-04-13 16:19 by 「初见」

并行运算、cool、

#19楼[楼主]　回复　引用
2012-04-16 08:50 by webabcd

@「初见」
是，而且到了4.5也会发扬光大

发表评论

昵称：　楚人游子

评论内容：

提交评论　注销　订阅评论

[Ctrl+Enter快捷键提交]

最新IT新闻:

· 创业公司应该招聘什么样的人?
· 池建强：每个人都该懂点写作
· 阿里云发布人工智能平台DTPAI
· 他才21岁，却要改变这个世界，因为大海被人弄脏了
· 升还是不升Windows 10 ，这是一个问题
» 更多新闻...

最新知识库文章:

· 关于烂代码的那些事（中）
· 关于烂代码的那些事（上）
· 作为码农，我们为什么要写作
· 今天你写了自动化测试吗
· 数学和编程
» 更多知识库文章...