目录

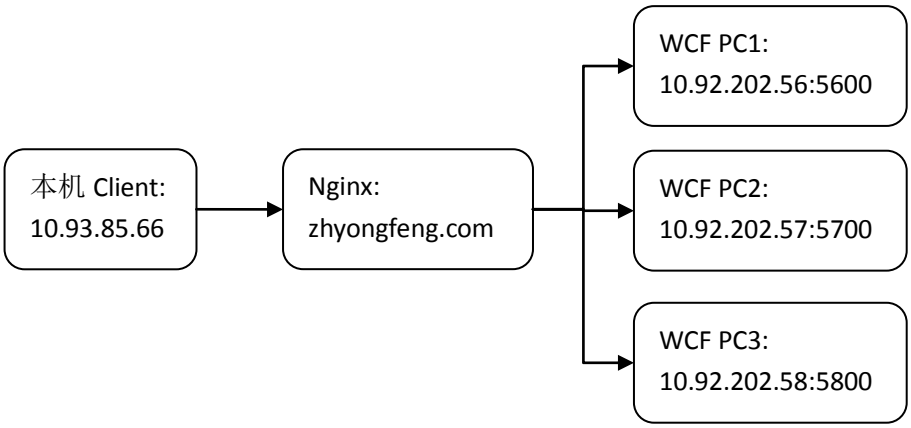
1 大概思路

- Nginx 集群 WCF 分布式局域网结构图
- 关于 WCF 的 BasicHttpBinding
- 编写 WCF 服务、客户端程序
- URL 保留项
- 部署 WCF 服务程序到局域网内 3 台 PC 机
- Nginx 集群配置搭建
- 启动 WCF 客户端程序
- 总结

2 Nginx 集群 WCF 分布式局域网结构图

关于 WCF 即可以寄宿于 IIS，也可以自我寄宿，本文采用的是自我寄宿方式。之所以采用自我寄宿方式，很大程度上，在一些特殊的场景，例如下载大文件（如几百 MB、1G 等）、图片、文档等，如果以 IIS 为宿主，可能会产生内存不够用。所以这里采用自我寄宿的方式为例子。

Nginx 集群除了在反向代理可以应用到，在 WCF 的处理上，也有很好的表现。以下是 Nginx 集群在 WCF 分布式的设计结构图：



3 关于 WCF 的 BasicHttpBinding

首先了解系统预定义绑定对不同安全模式的支持，具体参照以下该文：
[WCF 安全系列]绑定、安全模式与客户端凭证类型：总结篇
https://www.cnblogs.com/artech/archive/2011/05/28/Authentication_034.html

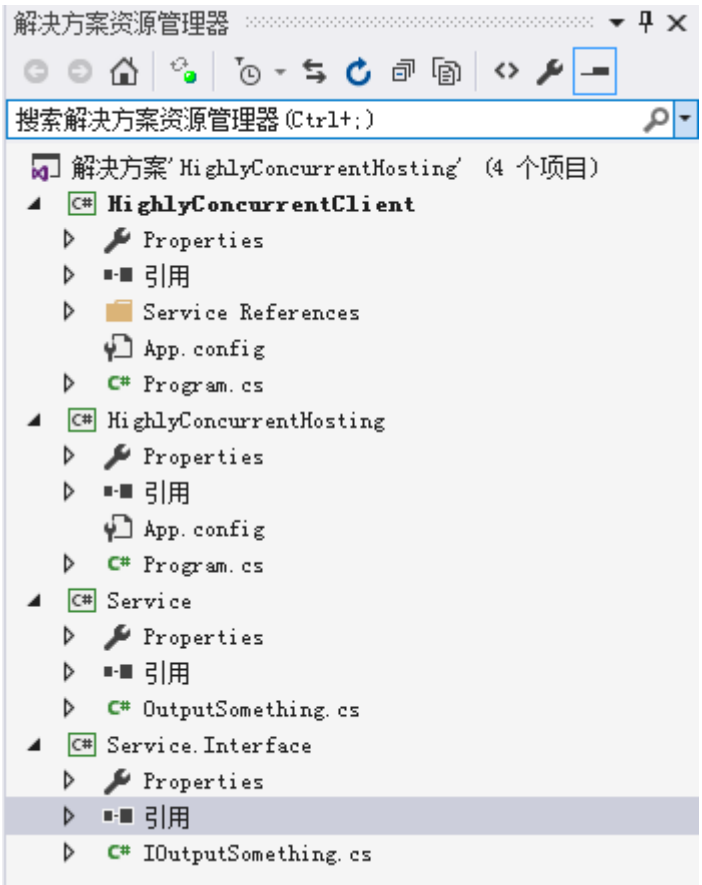
- 所有的绑定都可以不采用任何的安全传输机制，即支持 None 安全模式；

- BasicHttpBinding 的默认模式为 None，WS 相关的绑定默认模式为 Message，而局域网相关绑定的模式模式为 Transport；
- 除了 NetNamedPipeBinding，所有的绑定都支持 Message 安全模式；
- 对于所有支持 Message 模式的绑定，除了 NetMsmqBinding 都支持 Mixed 模式；
- 除了 WSDualHttpBinding，所有的绑定都支持 Transport 模式；
- 只有 BasicHttpBinding 支持 TransportCredentialOnly 模式；
- 只有 NetMsmqBinding 支持 Both 安全模式。

这里 WCF 的 Ningx 集群，主要用的是 BasicHttpBinding。表示一个绑定，Windows Communication Foundation (WCF) 服务可以使用此绑定配置和公开这样的终结点：这些终结点能够与基于 ASMX 的 Web 服务和客户端以及符合 WS-I Basic Profile 1.1 标准的其他服务进行通信。

4 编写 WCF 服务、客户端程序

- 解决方案的主要目录如下：



- WCF 服务程序

IOutputSomething.cs

```
using System.ServiceModel;
```

```
namespace Service.Interface
{
    [ServiceContract]
    public interface IOutputSomething
    {
        [OperationContract]
        string GetContentData(int i);

        [OperationContract]
        string GetIpAddress();
    }
}
```

OutputSomething.cs

```
using Service.Interface;
using System.Net;
using System.ServiceModel;

namespace Service
{
    [ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Multiple)]
    public class OutputSomething : IOutputSomething
    {
        /// <summary>
        ///  名称
        /// </summary>
        string threadNumber;

        readonly object thisLocak = new object();
        public string GetContentData(int i)
        {
            lock (thisLocak)
            {

                threadNumber = i.ToString() + " - " + "我是主机:" + GetIpAddress();
            }
            return string.Format("序列号{0},线程号{1}", i, threadNumber);
        }

        public string GetIpAddress()
        {
            string AddressIP = string.Empty;
            foreach (IPAddress _IPAddress in
Dns.GetHostEntry(Dns.GetHostName()).AddressList)
            {
```

```
                    if (_IPAddress.AddressFamily.ToString() == "InterNetwork")
                    {
                        AddressIP = _IPAddress.ToString();
                    }
                }
                return AddressIP;
            }
        }
}
```

Program.cs

```
using Service;
using System;
using System.ServiceModel;

namespace HighlyConcurrentHosting
{
    class Program
    {
        static void Main(string[] args)
        {
            using (ServiceHost host = new ServiceHost(typeof(OutputSomething)))
            {
                host.Opened += delegate
                {
                    Console.WriteLine(host.Description.Endpoints[0].Address.Uri + "已经启动，按任意键终止服务！");
                };

                host.Open();
                Console.Read();
            }
        }
    }
}
```

- 客户端程序

Program.cs

```
using HighlyConcurrentClient.HighlyConcurrentService;
using System;
using System.Net;

namespace HighlyConcurrentClient
{
    class Program
    {
```

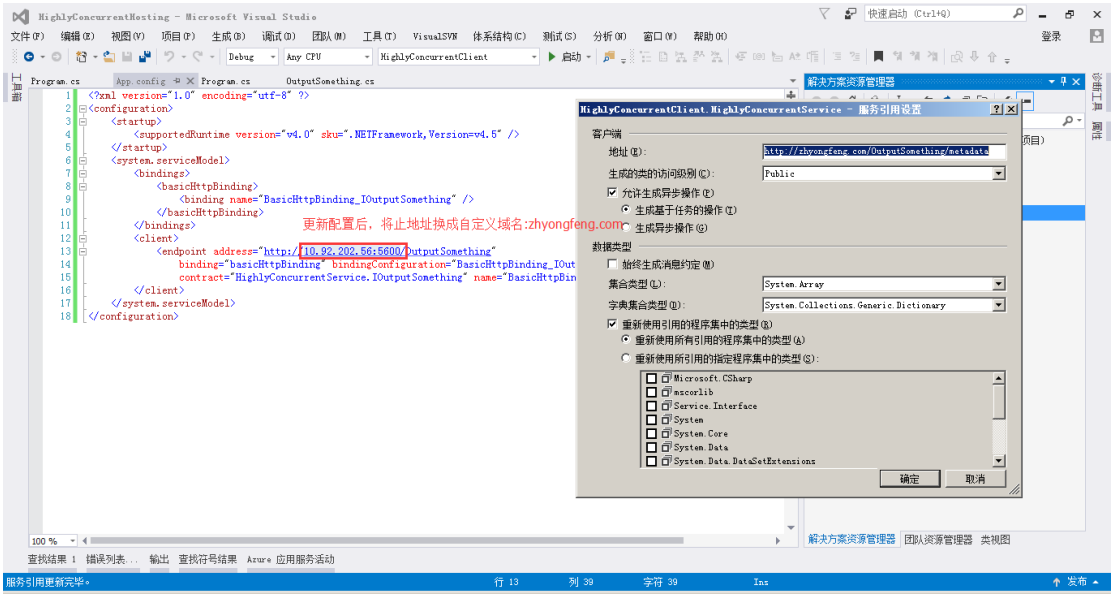
```
        static void Main(string[] args)
        {

            string AddressIP = string.Empty;
            foreach (IPAddress _IPAddress in
Dns.GetHostEntry(Dns.GetHostName()).AddressList)
            {
                if (_IPAddress.AddressFamily.ToString() == "InterNetwork")
                {
                    AddressIP = _IPAddress.ToString();
                }
            }
            Console.WriteLine("本机 IP 是：" + AddressIP);
            using (OutputSomethingClient proxy = new OutputSomethingClient())
            {
                for (int i = 0; i < 20; i++)
                {
                    Console.WriteLine(proxy.GetContentData(i));
                }
            }
            Console.Read();
        }
    }
}
```

客户端添加服务引用后，Address 可能是某一台 PC 机的 IP 地址（例如：address="http://10.92.202.56:5600/OutputSomething"）这是需要修改为以下 Nginx 的地址 address="http://zhyongfeng.com/OutputSomething"，配置如下：

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
    </startup>
    <system.serviceModel>
        <bindings>
            <basicHttpBinding>
                <binding name="BasicHttpBinding_IOutputSomething" />
            </basicHttpBinding>
        </bindings>
        <client>
            <endpoint address="http://zhyongfeng.com/OutputSomething"
                binding="basicHttpBinding"
bindingConfiguration="BasicHttpBinding_IOutputSomething"
                contract="HighlyConcurrentService.IOutputSomething"
name="BasicHttpBinding_IOutputSomething" />
```

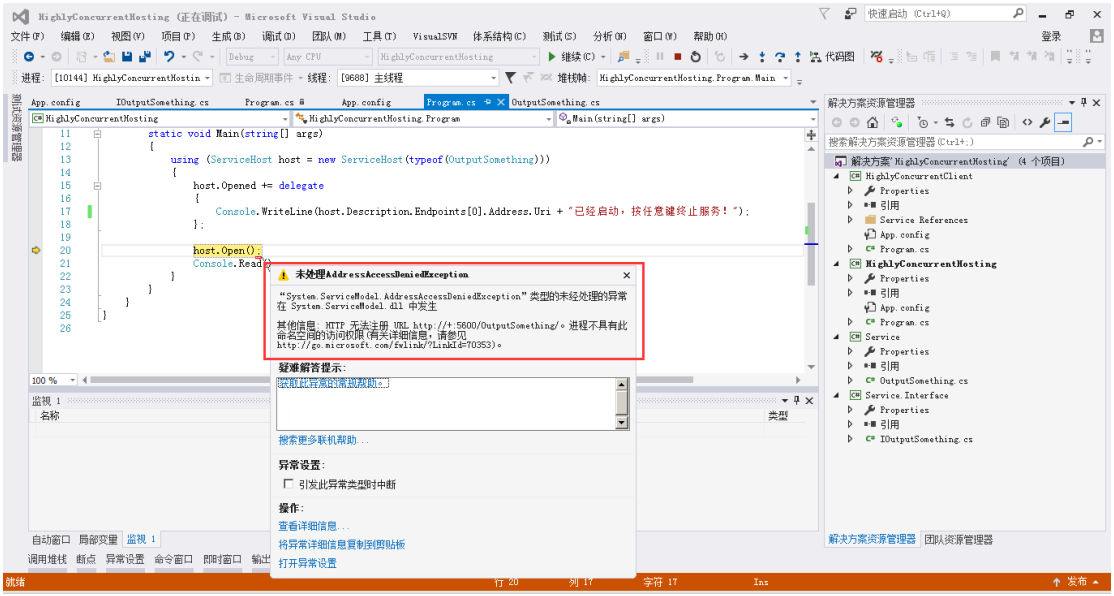
```
        </client>
    </system.serviceModel>
</configuration>
```

即如图所示：



5 URL 保留项

在默认的操作系统配置中，Windows Communication Foundation (WCF) 为端口 80 创建可全局访问的保留项，使所有用户都能够运行应用程序，在该应用程序中使用双向 HTTP 绑定来进行双工通信。



返回了：

“System.ServiceModel.AddressAccessDeniedException”类型的未经处理的异常在 System.ServiceModel.dll 中发生

其他信息: HTTP 无法注册 URL http://+:5600/OutputSomething/。进程不具有此命名空间的访问权限(有关详细信息，请参见 http://go.microsoft.com/fwlink/?LinkId=70353)。

以管理员方式运行 C:\windows\System32\cmd.exe：

```
netsh http add urlacl url=http://+:5600/ user="\Everyone"
netsh http add iplisten ipaddress=0.0.0.0:5600
```

防火墙的入站规则：

```
netsh advfirewall firewall add rule name="5600 端口" dir=in action=allow protocol=TCP localport=5600
```

url 保留项删除

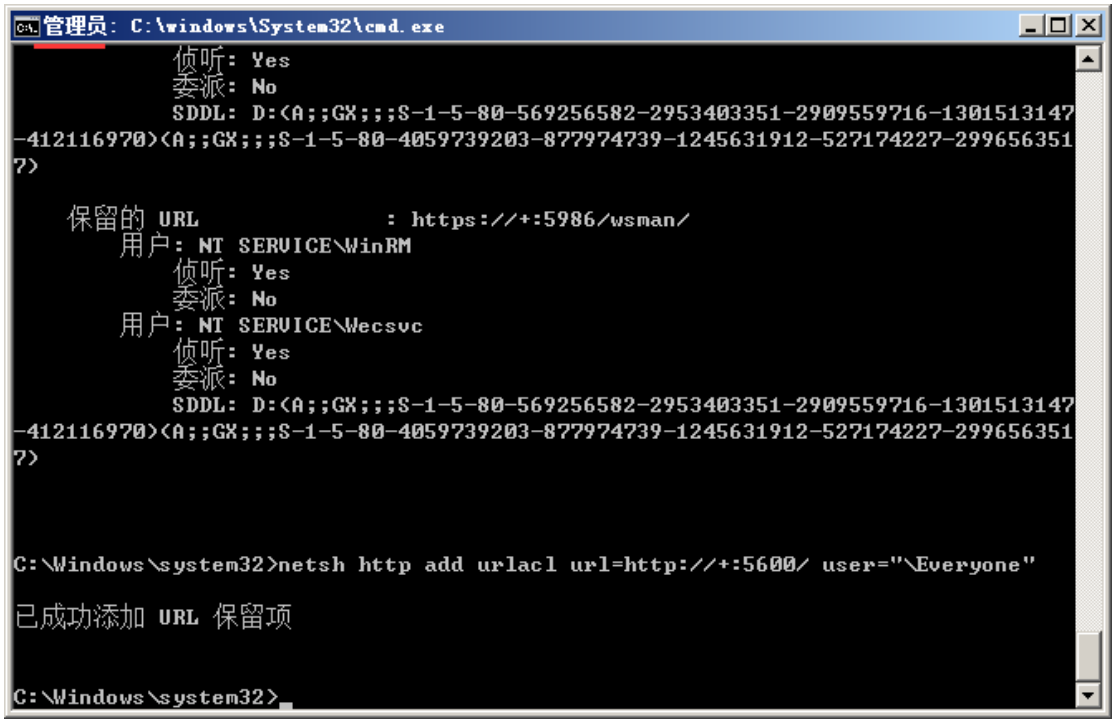
```
netsh http delete urlacl url=http://+:5600/
```

url 保留项显示

```
netsh http show urlacl
```

这里主要使用如下添加 URL 保留项即可：

```
netsh http add urlacl url=http://+:5600/ user="\Everyone"
```

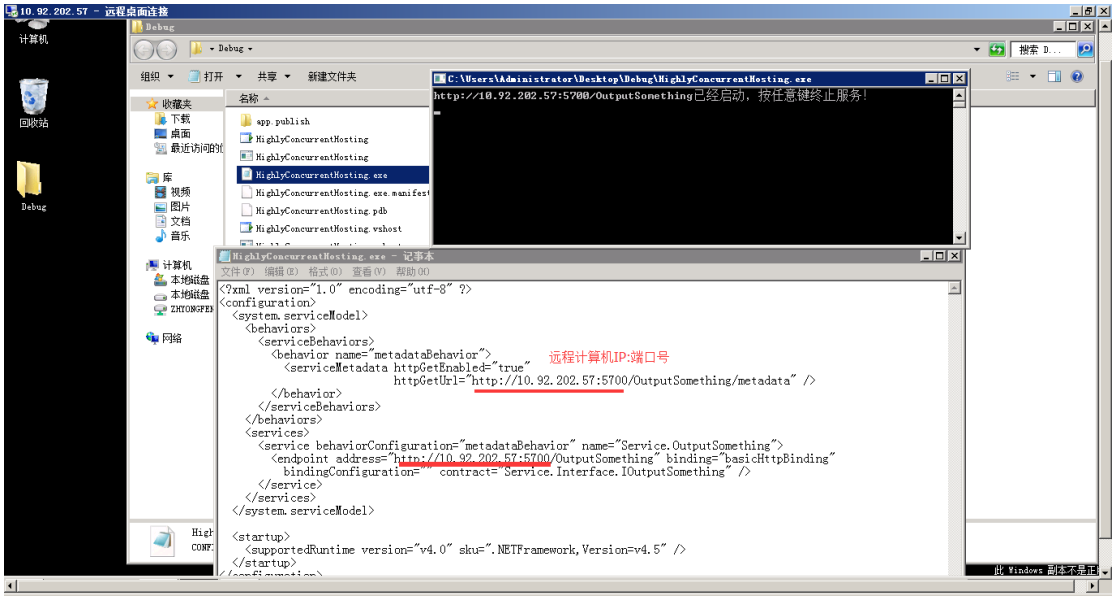


6 部署 WCF 服务程序到局域网内 3 台 PC 机

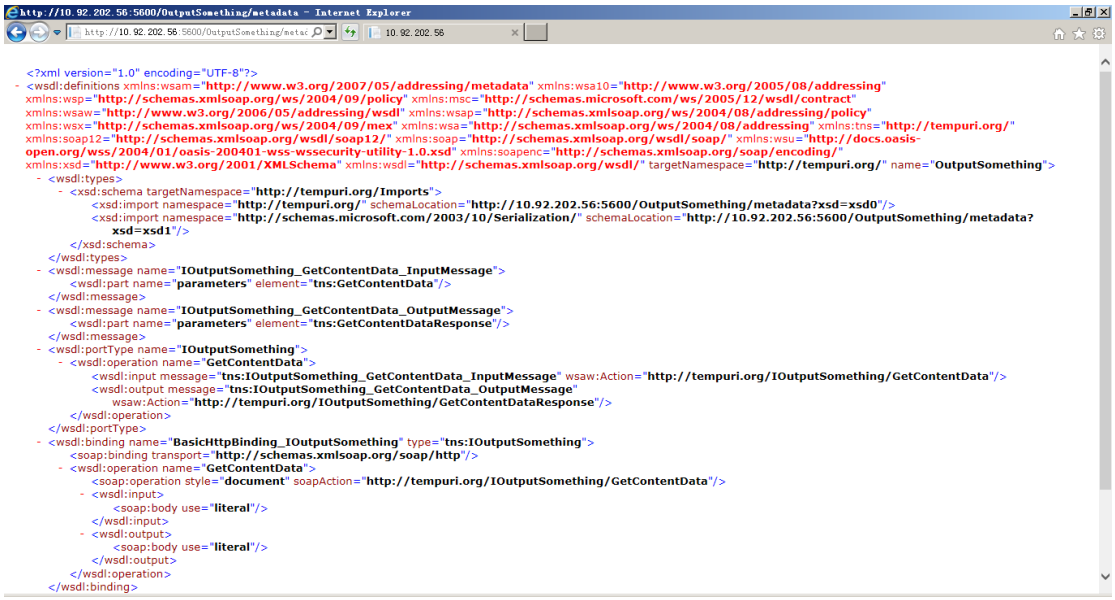
远程进行部署 WCF 服务程序时，需要将它的 config 配置文件，重新定位 address，将默认的 IP 地址 127.0.0.1:5600 修改为远程计算机的 IP 地址：

10.92.202.56:5600、10.92.202.57:5700、10.92.202.58:5800

然后启动远程计算机的 WCF 服务程序，运行效果如下：



本机 IE 上访问 WCF 服务端的运行效果：



7 Nginx 集群配置搭建

通过自主义域名 zhyongfeng.com：80 端口进行负载均衡集群访问，则访问 C:\Windows\System32\drivers\etc\hosts，添加下列“本机 IP 自定义的域名”：

```
10.93.85.66    zhyongfeng.com
```

针对 WCF 部署的多台 PC 机配置（设置了 proxy_connect_timeout 为 10s，如果其中一台机 down 掉了，可以转发到另一台机器）如下：

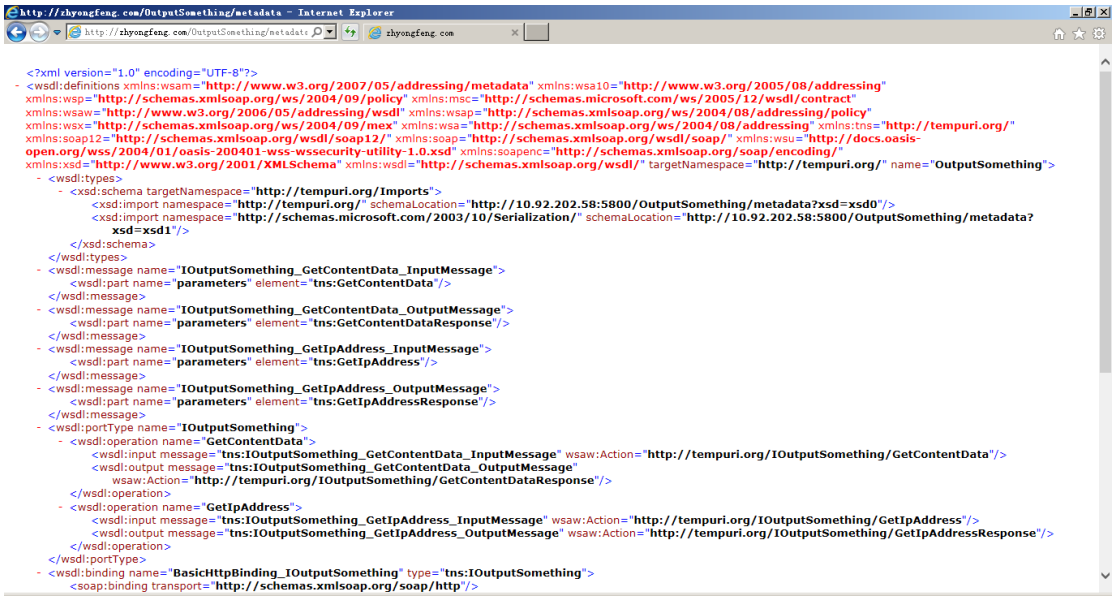
```
worker_processes    1;
events {
    worker_connections    1024;
}
```

```
http {
    include       mime.types;
    default_type  application/octet-stream;
    sendfile        on;
    keepalive_timeout  65;

    upstream zhyongfeng.com {
        server    10.92.202.56:5600;
        server    10.92.202.57:5700;
        server    10.92.202.58:5800;
    }
    server {
        listen       80;
        server_name  zhyongfeng.com;
        location / {
            proxy_pass    http://zhyongfeng.com;
            proxy_connect_timeout       10s;
        }
    }
}
```

运行 CMD：

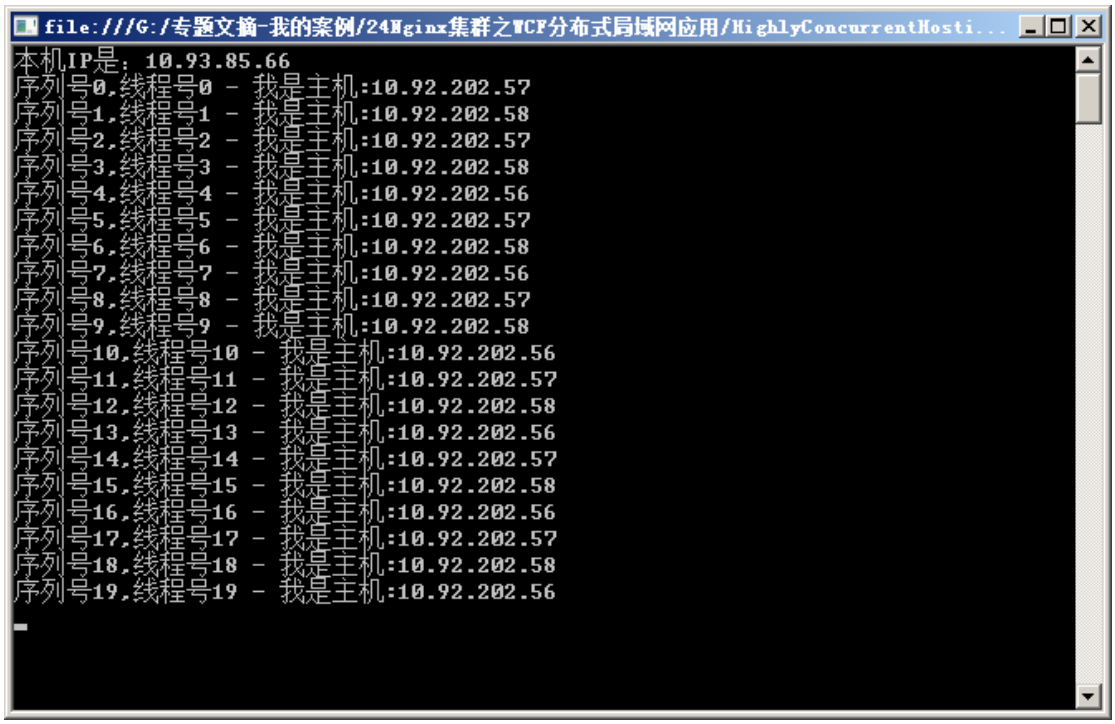
```
D:\DTLDownLoads\nginx-1.10.2>start nginx

D:\DTLDownLoads\nginx-1.10.2>nginx -s reload
```

访问 WCF 服务端：http://zhyongfeng.com/OutputSomething/metadata，运行结果：

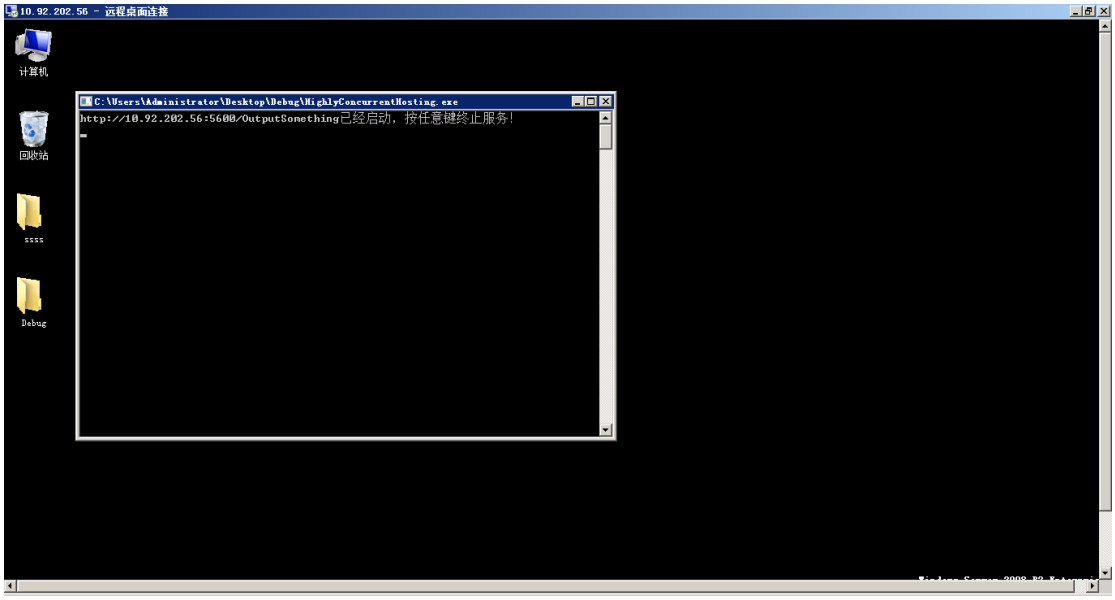


8 启动 WCF 客户端程序

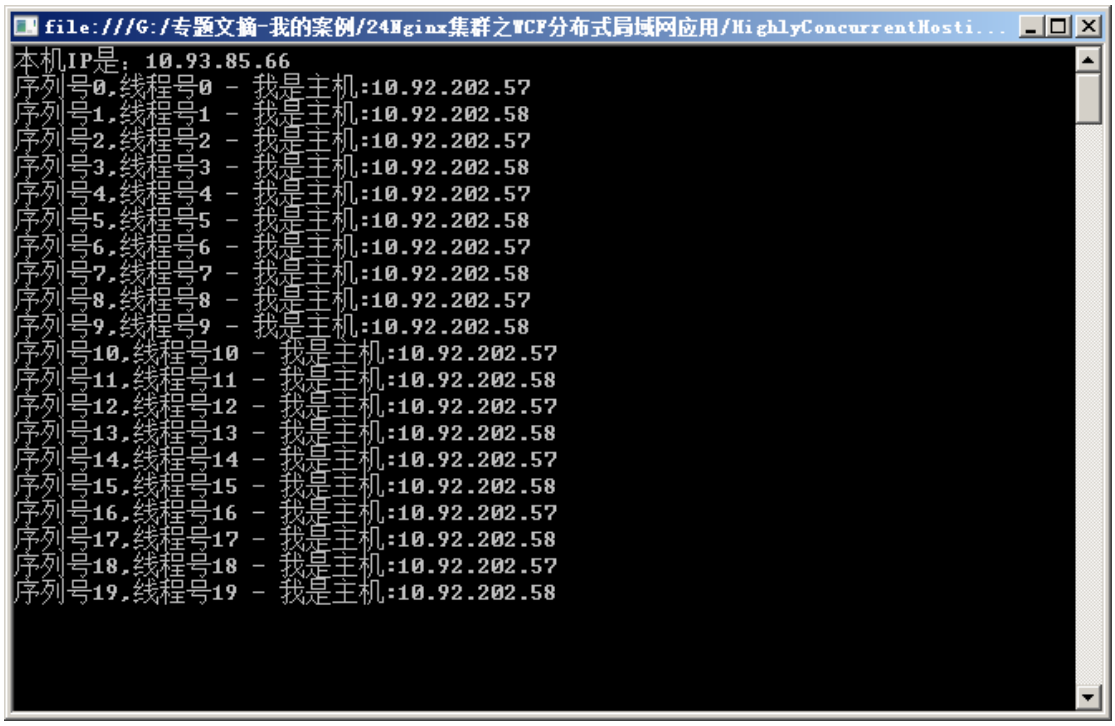
启动 WCF 客户端程序，运行效果图如下：



远程桌面关掉其中一台 10.92.202.56:5600 的 PC 机：



重新启动 WCF 客户端程序，因为 Nginx 配置文件设置了 proxy_connect_timeout 为 10s，则关闭的 PC 机 10.92.202.56:5600 在 10s 后会将它的消息转发给 10.92.202.57:5700，继续由其它 2 台 PC 机执行：



9 总结

WCF是由微软开发的一系列支持数据通信的应用程序框架，通过开源框架Nginx的结合，能够有更多的扩展性。Nginx 结合 WCF 对局域网内的布局有很大关系，通过 WCF 整合报表服务器、邮件服务器、文档服务器等，WCF 原来就整合了原有的 windows 通讯的 .net Remoting，WebService，Socket 的机制，Nginx 让具备分布式功能的 WCF 更加强大了。

源代码下载：
http://download.csdn.net/download/ruby_matlab/10122670

PDF 下载：
Nginx 集群之 WCF 分布式局域网应用.pdf