目录

1 大概思路

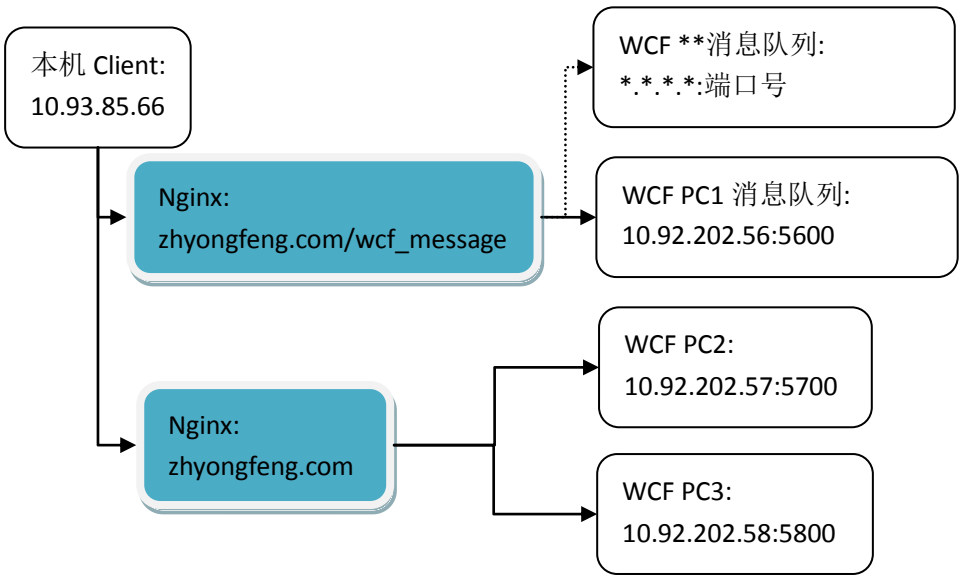
- Nginx 集群之 WCF 分布式消息队列
- MSMQ 消息队列
- 编写 WCF 服务、客户端程序
- 服务器安装 MSMQ
- 部署 WCF 服务程序到局域网内 1 台 PC 机
- Nginx 集群配置搭建
- 运行结果
- 总结

2 Nginx 集群之 WCF 分布式消息队列

针对 WCF 分布式消息队列 MSMQ 大大提高了处理能力，无论是发送方还是接收方都不用等待对方返回成功消息，但是不适合 Client 与 Server 端的实时交互。WCF 分布式消息队列，在处理日志方面，效果还是很显著的。

当然，针对消息队列的处理技术，有很多种，例如：ActiveMQ、RabbitMQ、ZeroMQ、Kafka、MetaMQ、RocketMQ。本文使用的是微软自带的消息队列 MSMQ，结合 WCF 在 Ningx 集群的环境下，创建一个类似日志型或邮件型的 WCF 服务。

以下是 WCF 分布式消息队列的架构：



3 MSMQ 消息队列

消息队列 MSMQ 提供更高程序的非连接通信支持，因为它支持离线通信方式。消息的发送者在离线方式下进行，离线通信方式让进行消息交换的参与者变成完全独立的两个应用，它们之间唯一的纽带就是消息队列。

如果借助 MSMQ 提供的通信方式，无论出现多高的负载，只要抵达的消息累积量不超过设定的限额，MSMQ 就完全可以按照自己的节奏来进行处理。将高负载的请求延后到低负载的时候进行处理，很好地解决了负载上面的问题。

MSMQ 中主要有两个概念。

一个是消息 Message：Message 是通信双方需要传递的消息，它可以是文本、图片、视频等。消息包含发送和接收者的标识，只有指定的用户才能取得消息。

队列	描述
公共队列	在整个消息队列网络中复制，有可能由网络连接的所有站点访问
专用队列	不在整个网络中发布，它们仅在所驻留的本地计算机上可用，专用队列只能由知道队列的完整路径名称或标签的应用程序访问
日志队列	包含确认在给定“消息队列中发送的消息回执消息”
响应队列	包含目标应用程序接收到消息时返回给发送应用程序的响应消息，包括机器日志队列、机器死信队列和机器事务死信队列

4 编写 WCF 服务、客户端程序

- WCF 服务程序

Program.cs

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

namespace MessageDealingHosting
{
    class Program
        {
            static void Main(string[] args)
            {
                using (ServiceHost host = new ServiceHost(typeof(PublicMessageQueue)))
                {
                    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();
                    }
                }
            }
}
```

PublicMessageQueue.cs

```
using System;
using Service.Interface;

namespace Service
{
    public partial class PublicMessageQueue : IPublicMessageQueue
    {
        public void SendMessage(string msg)
        {
            Console.WriteLine(string.Format("消息队列开始处理，输入{0}成功", msg));
        }
    }
}
```

IPublicMessageQueue.cs

```
using System;
using Service.Interface;

namespace Service
{
    public partial class PublicMessageQueue : IPublicMessageQueue
    {
        public void SendMessage(string msg)
        {
            Console.WriteLine(string.Format("消息队列开始处理，输入{0}成功", msg));
        }
    }
}
```

服务端配置文件：

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
```

```
    <behaviors>
      <serviceBehaviors>
        <behavior name="MessageQueueBehavior">
          <serviceMetadata httpGetEnabled="true" />
          <serviceDebug includeExceptionDetailInFaults="true" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <bindings>
      <netMsmqBinding>
        <binding name="MessageQueueBinding" exactlyOnce="false">
          <security mode="None"></security>
        </binding>
      </netMsmqBinding>
    </bindings>
    <services>
      <service                                    behaviorConfiguration="MessageQueueBehavior"
name="Service.PublicMessageQueue">
        <endpoint                    address="net.msmq://10.92.202.56/private/wcf_message"
binding="netMsmqBinding" bindingConfiguration="MessageQueueBinding"
          contract="Service.Interface.IPublicMessageQueue" />
        <host>
          <baseAddresses>
            <add baseAddress="http:// 10.92.202.56:5600/wcf_message" />
          </baseAddresses>
        </host>
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

- 客户端程序

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using MessageDealingClient.MessageDealingService;
using System.ServiceModel;

namespace MessageDealingClient
{
    class Program
    {
        static void Main(string[] args)
```

```
        {
            using (ChannelFactory<IPublicMessageQueue> channelFactory = new
ChannelFactory<IPublicMessageQueue>("NetMsmqBinding_IPublicMessageQueue"))
            {
                IPublicMessageQueue proxyServer = channelFactory.CreateChannel();
                proxyServer.SendMessage("hello world");
                Console.WriteLine("客户端先启动，将消息传到消息队列，若 WCF 服务启
动，则会输出 hello world");
            }
            Console.Read();
        }
    }
}
```

客户端配置文件：

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <system.serviceModel>
        <bindings>
            <netMsmqBinding>
                <binding                    name="NetMsmqBinding_IPublicMessageQueue"
exactlyOnce="false">
                    <security mode="None" />
                </binding>
            </netMsmqBinding>
        </bindings>
        <client>
            <endpoint address="net.msmq://10.92.202.56/private/wcf_message"
                binding="netMsmqBinding"
bindingConfiguration="NetMsmqBinding_IPublicMessageQueue"
                contract="MessageDealingService.IPublicMessageQueue"
name="NetMsmqBinding_IPublicMessageQueue" />
        </client>
    </system.serviceModel>
</configuration>
```

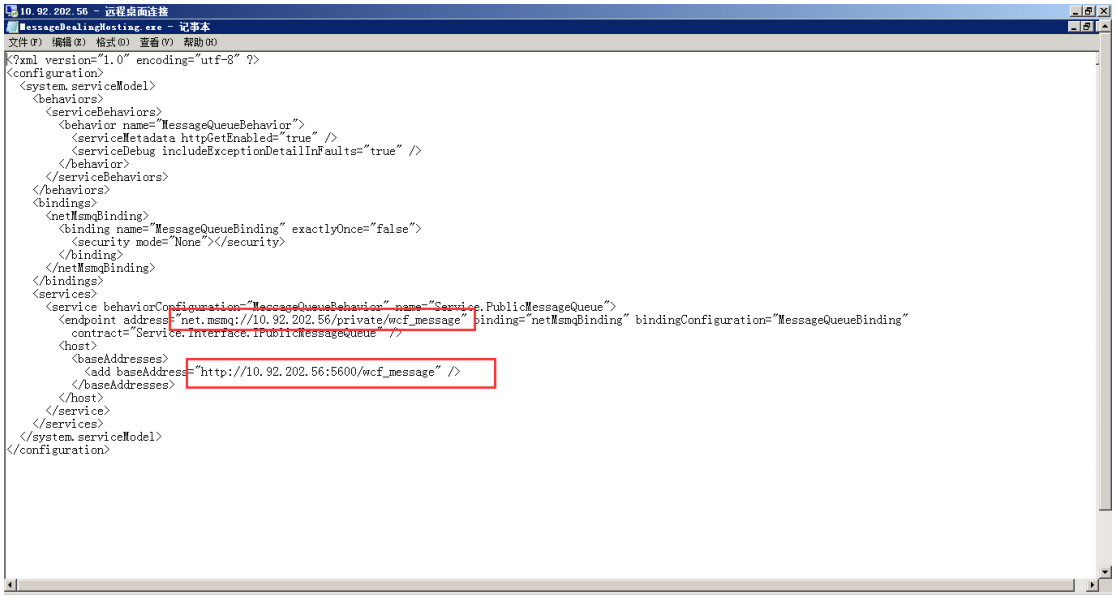
5 服务器安装 MSMQ

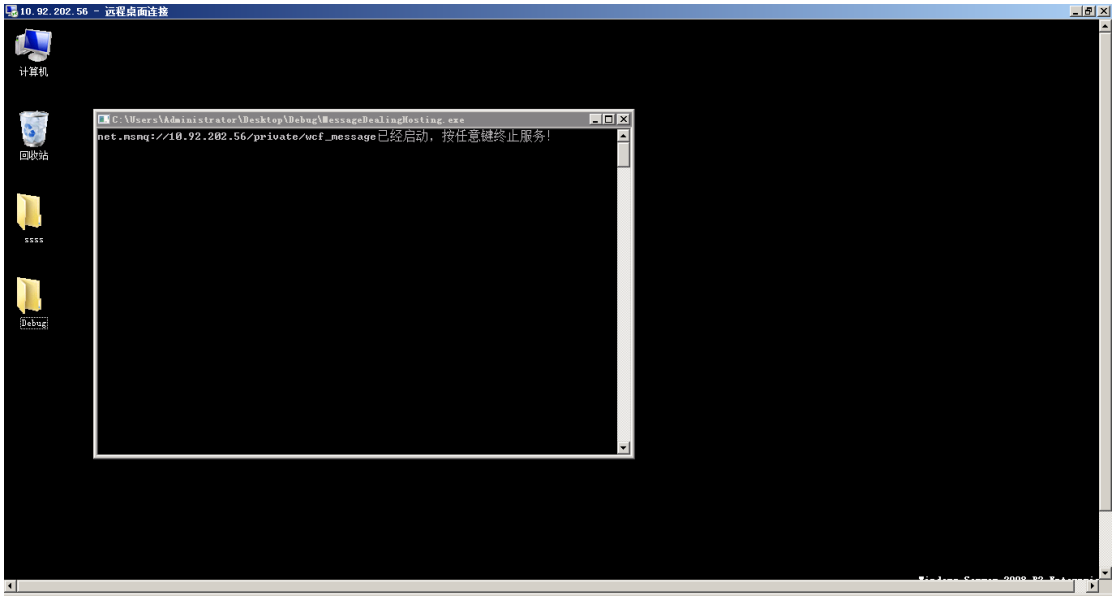
- 打开“控制面板”
- 单击“程序”，然后在“程序和功能”下单击“打开或关闭 Windows 功能
- 展开“Microsoft Message Queue (MSMQ) 服务器”，展开“Microsoft Message Queue (MSMQ) 服务器核心”，然后选中对应于以下要安装的“消息队列”功能的复选框
  - MSMQ Active Directory 域服务集成（用于加入域的计算机）。

- MSMQ HTTP 支持。

6 部署 WCF 服务程序到局域网内 1 台 PC 机

远程部署 WCF 服务端程序到 PC 机





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
```

使用 Nginx 匹配原则针对 WCF 部署的 1 台 PC 机配置如下：

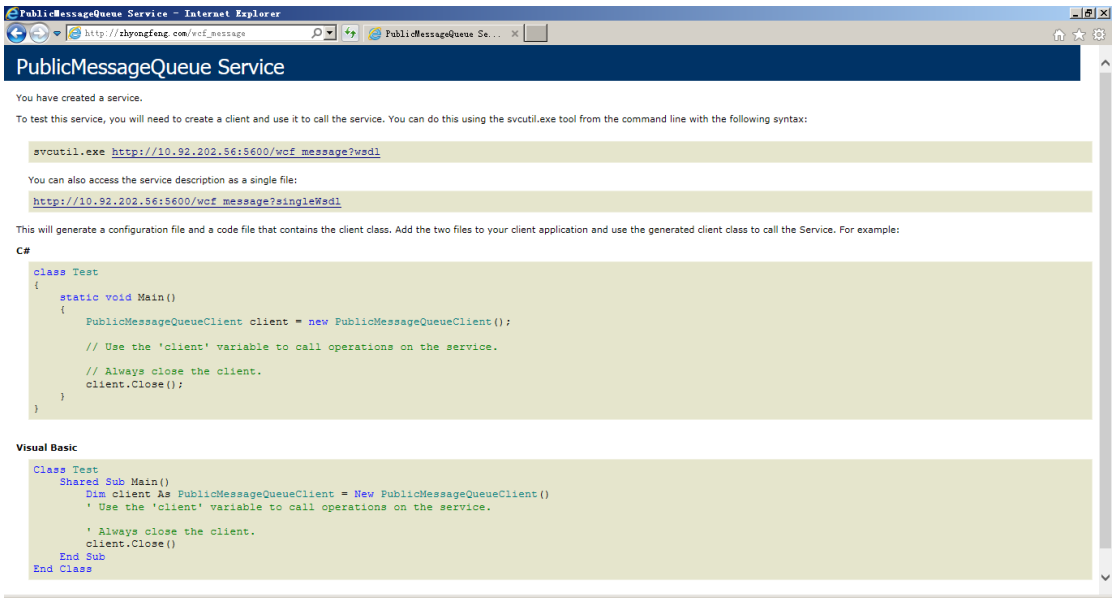
```
worker_processes    1;
events {
    worker_connections    1024;
}
http {
    include            mime.types;
    default_type    application/octet-stream;
    sendfile            on;
    keepalive_timeout    65;

    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;
            }
        location /wcf_message{
                proxy_pass      http://10.92.202.56:5600/wcf_message;
                proxy_connect_timeout           10s;
            }
        }
}
```

运行 CMD：

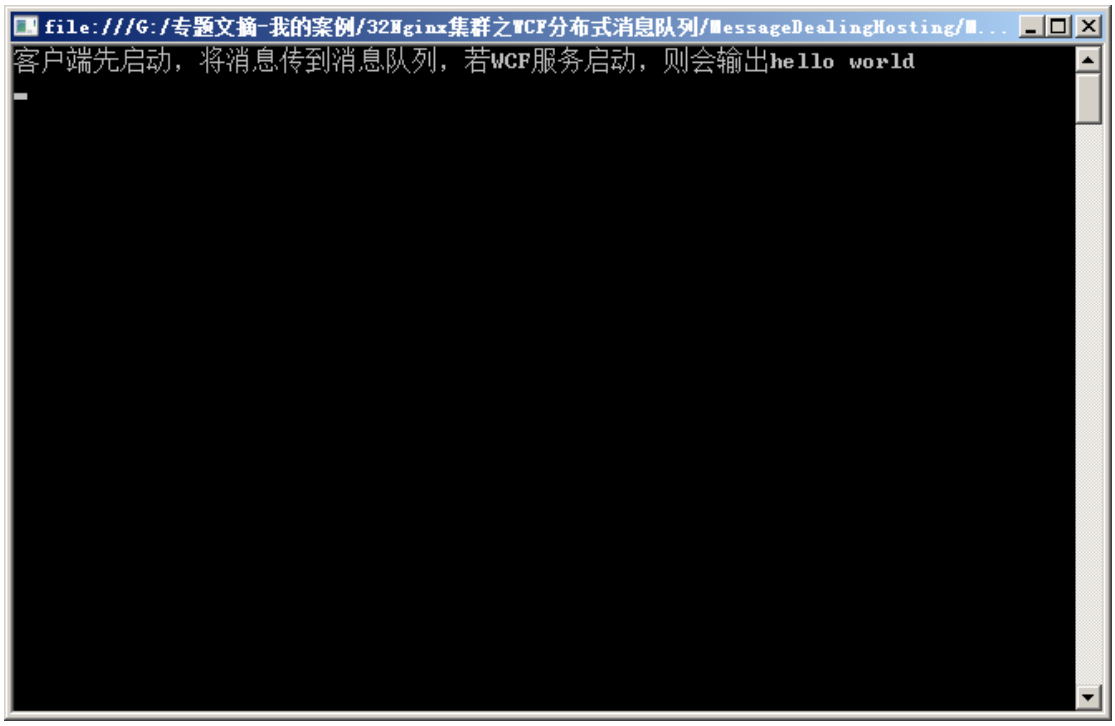
```
D:\DTLDownLoads\nginx-1.10.2>start nginx

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

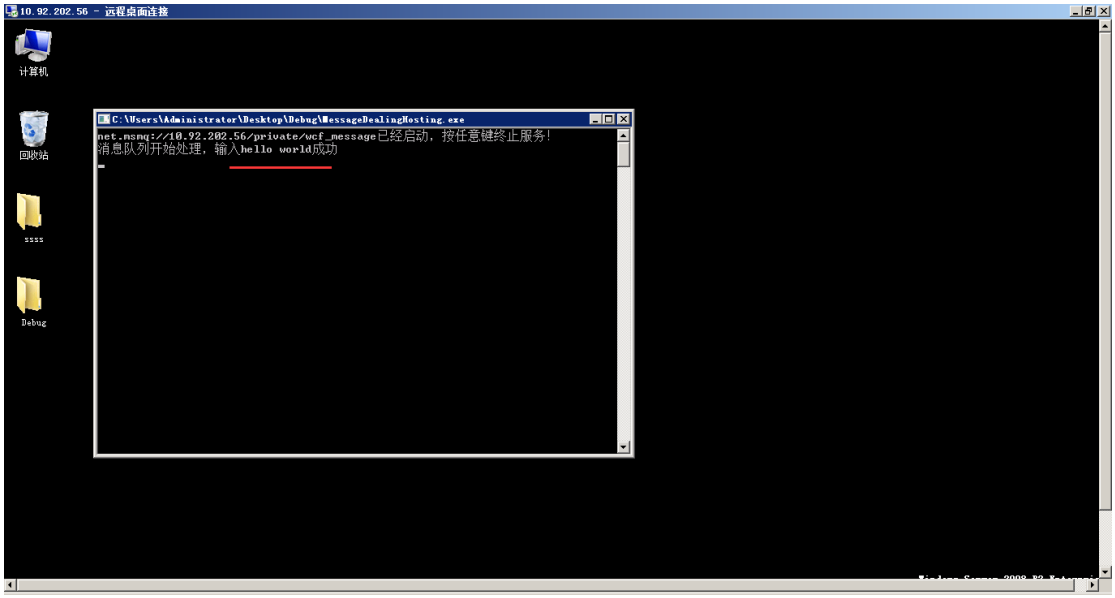
访问 WCF 服务端：http://zhyongfeng.com/wcf_message，运行结果：



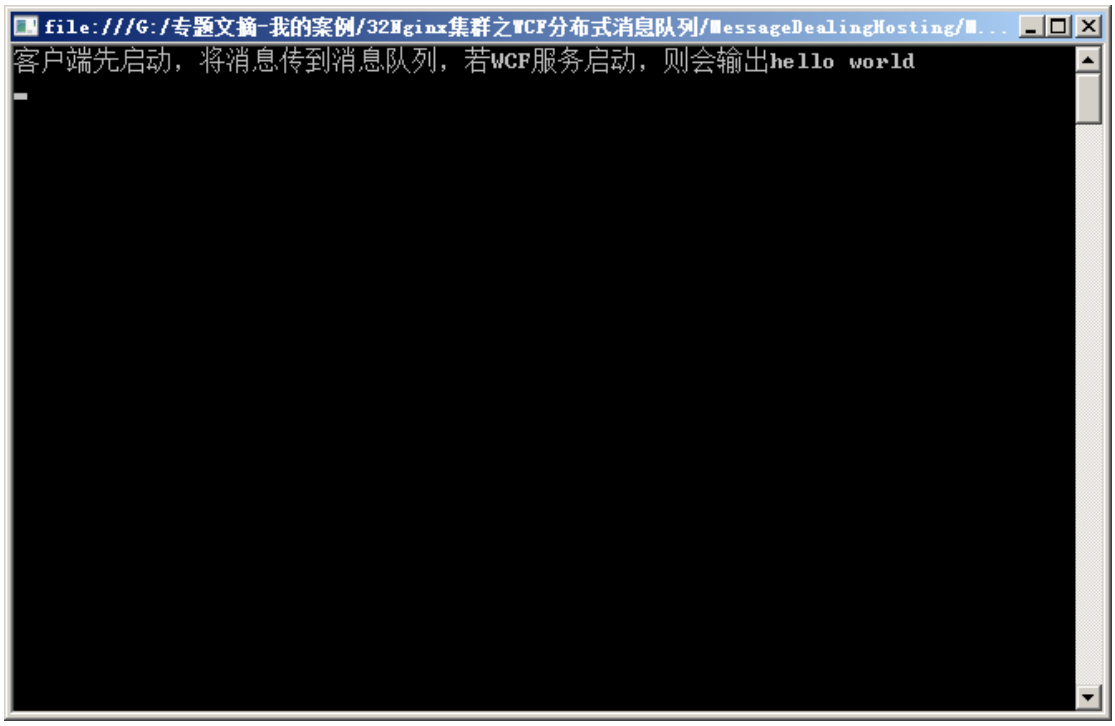
8 运行结果

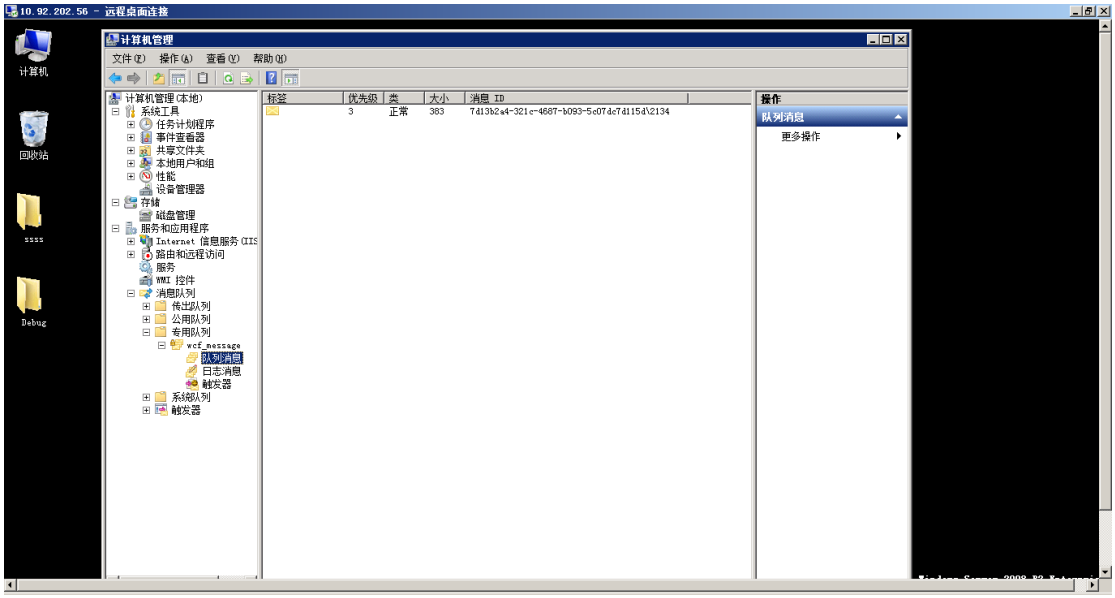
启动远程 WCF 服务端，同时启动 WCF 客户端，结果如下：



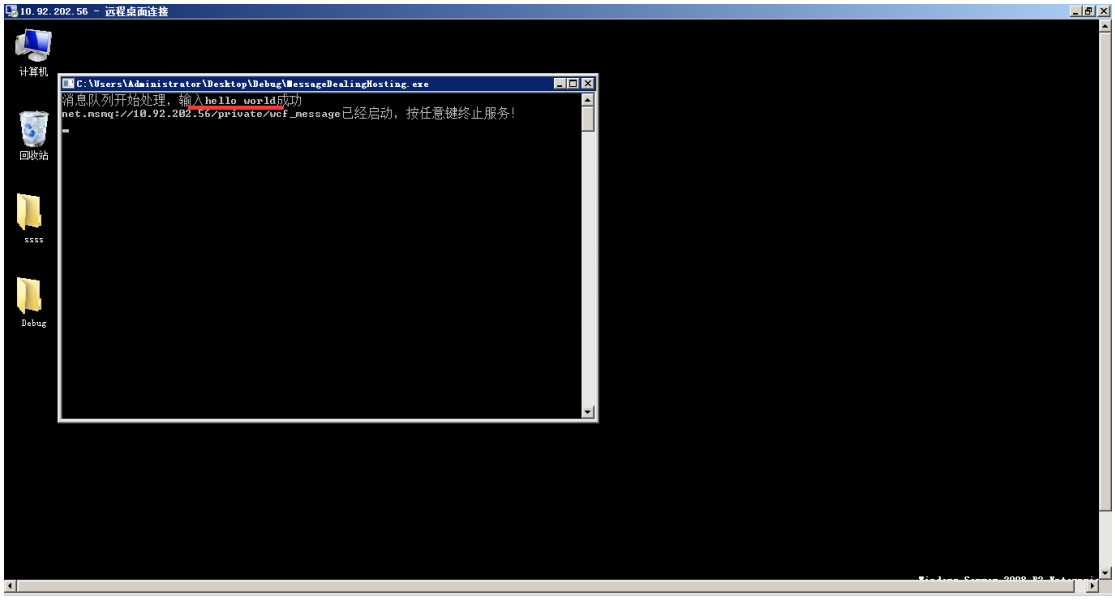


关闭远程 WCF 服务端，启动 WCF 客户端，结果如下：





再启动 WCF 服务端，结果如下：



9 总结

基于 WCF 分布式消息队列，可以在一些客户端并不需要服务端响应的场景上应用。消息队列在异步处理上有巨大优势，是一项可选择性的进程间的通信，能够保持进程之间通信的稳定性。

源代码下载：
http://download.csdn.net/download/ruby_matlab/10134565

PDF 下载：
Nginx 集群之 WCF 分布式消息队列.pdf