目录

1 大概思路.....1
2 Nginx 集群之 WCF 大文件上传及下载.....1
3 BasicHttpBinding 相关配置解析.....2
4 编写 WCF 服务、客户端程序.....3
5 URL 保留项.....8
6 部署 WCF 服务程序到局域网内 1 台 PC 机.....8
7 Nginx 集群配置搭建.....9
8 WCF 客户端程序的运行结果.....11
9 总结.....13

1 大概思路

- Nginx 集群之 WCF 大文件上传及下载
- BasicHttpBinding 相关配置解析
  transferMode、messageEncoding、maxReceivedMessageSize、receiveTimeout、sendTimeout
- 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 1 台 PC 机
- Nginx 集群配置搭建
- WCF 客户端程序的运行结果
- 总结

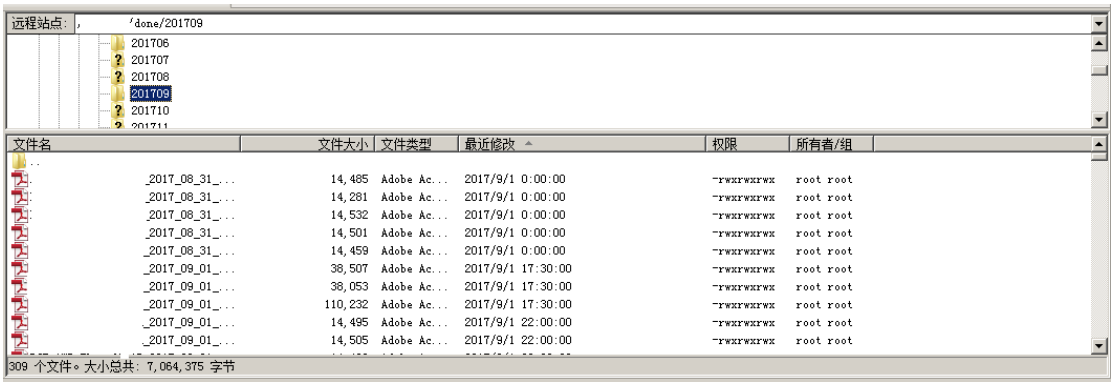
2 Nginx 集群之 WCF 大文件上传及下载

Nginx 的匹配规则，很容易帮助我们划分 WCF 服务的网段，从而实现企业数据信息系统多区域划分，如小数据的微服务、传输数据文件的服务、即时通信服务、或者邮件服务，相当于构建了一条企业内部信息化的数据总线（DataBus）。

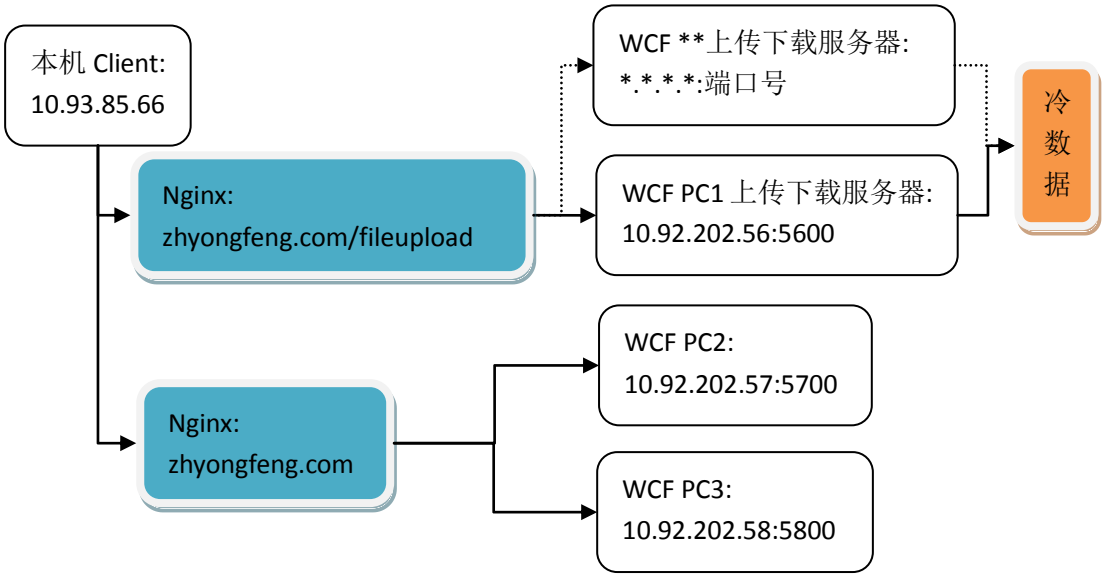
以下是本文讲述的主要结构图：

客户端以 BasicHttpBinding 访问 Nginx 的域名 zhyongfeng.com/fileupload，然后 Nginx 进行负载均衡，将消息分发到后端任意一台 WCF 上传下载服务器的 PC 机，然后进行上传文件至“冷数据”处，又从“冷数据”处下载文件。

针对“冷数据”可以划分目录，建立单独的 FTP 服务器及 WCF 服务器，进行操作处理。如下图所示：



以下是 WCF 上传下载服务器的架构：



3 BasicHttpBinding 相关配置解析

basicHttpBinding 相关配置，具体参考：
https://msdn.microsoft.com/zh-cn/library/system.servicemodel.basichttpbinding.aspx
以下是主要应用到的配置

transferMode	获取或设置一个值，该值指示是通过缓冲处理还是流处理来发送消息。（继承自 HttpBindingBase。）
messageEncoding	获取或设置是使用 MTOM 还是文本对 SOAP 消息进行编码。
maxReceivedMessageSize	获取或设置最大大小，以字节为单位，可以使用此绑定配置的通道上接收的消息。（继承自 HttpBindingBase。）
receiveTimeout	获取或设置连接在撤消之前保持非活动状态的最大时间间隔，在此时间间隔内未接收任何应用程序消息。（继承自 Binding。）
sendTimeout	获取或设置在传输引发异常之前可用于完成写入操作的时间间隔。（继承自 Binding。）

针对 MTOM 编码和异步调用的性能改善，可以参照论文：
《利用 MTOM 编码和异步调用改进流传输的性能.pdf》

4 编写 WCF 服务、客户端程序

- WCF 服务程序

Program.cs

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

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

FileTransferOperation.cs

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

namespace Service
{
    [ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Multiple)]
    public class FileTransferOperation : IFileTransferOperation
    {
        /// <summary>
        /// 上传文件
        /// </summary>
        /// <param name="remoteFile"></param>
```

```
        public void UploadLoad(RemoteFileInfo remoteFile)
        {
            StreamToFile(remoteFile);
        }

        /// <summary>
        /// 写文件
        /// </summary>
        /// <param name="remoteFile"></param>
        public void StreamToFile(RemoteFileInfo remoteFile)
        {
            string fileFullPath = Path.Combine(System.Environment.CurrentDirectory, remoteFile.FileName);
            Stream sourceStream = remoteFile.FileByteStream;
            if (sourceStream == null) { return; }
            if (!sourceStream.CanRead) { return; }
            //创建文件流，读取流中的数据生成文件
            using (FileStream fs = new FileStream(fileFullPath, FileMode.Create, FileAccess.Write, FileShare.None))
            {
                const int bufferLength = 4096;
                byte[] myBuffer = new byte[bufferLength];//数据缓冲区
                int count;
                while ((count = sourceStream.Read(myBuffer, 0, bufferLength)) > 0)
                {
                    fs.Write(myBuffer, 0, count);
                }
                fs.Close();
                sourceStream.Close();
            }
        }

        /// <summary>
        /// 下载文件
        /// </summary>
        /// <param name="fileName"></param>
        /// <returns></returns>
        public Stream DownLoad(string fileName)
        {
            string fileFullPath = Path.Combine(System.Environment.CurrentDirectory, fileName);
            if (!File.Exists(fileFullPath))//判断文件是否存在
            {
                return null;
```

```
                }
                try
                {
                    Stream myStream = File.OpenRead(fileFullPath);
                    return myStream;
                }
                catch { return null; }
            }
        }
}
```

服务端配置文件：

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior name="FileTransferBehavior">
          <serviceMetadata httpGetEnabled="true" />
          <serviceDebug includeExceptionDetailInFaults="true" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <bindings>
      <basicHttpBinding>
        <binding name="FileTransferdBinding" receiveTimeout="00:30:00"
sendTimeout="00:30:00" maxReceivedMessageSize="6442450944" transferMode="Streamed"
            messageEncoding="Mtom" />
      </basicHttpBinding>
    </bindings>
    <services>
      <service behaviorConfiguration="FileTransferBehavior"
name="Service.FileTransferOperation">
        <endpoint binding="basicHttpBinding" bindingConfiguration="FileTransferdBinding"
            contract="Service.Interface.IFileTransferOperation" />
        <host>
          <baseAddresses>
            <add baseAddress="http://127.0.0.1:5600/fileupload" />
          </baseAddresses>
        </host>
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

- 客户端程序

Program.cs

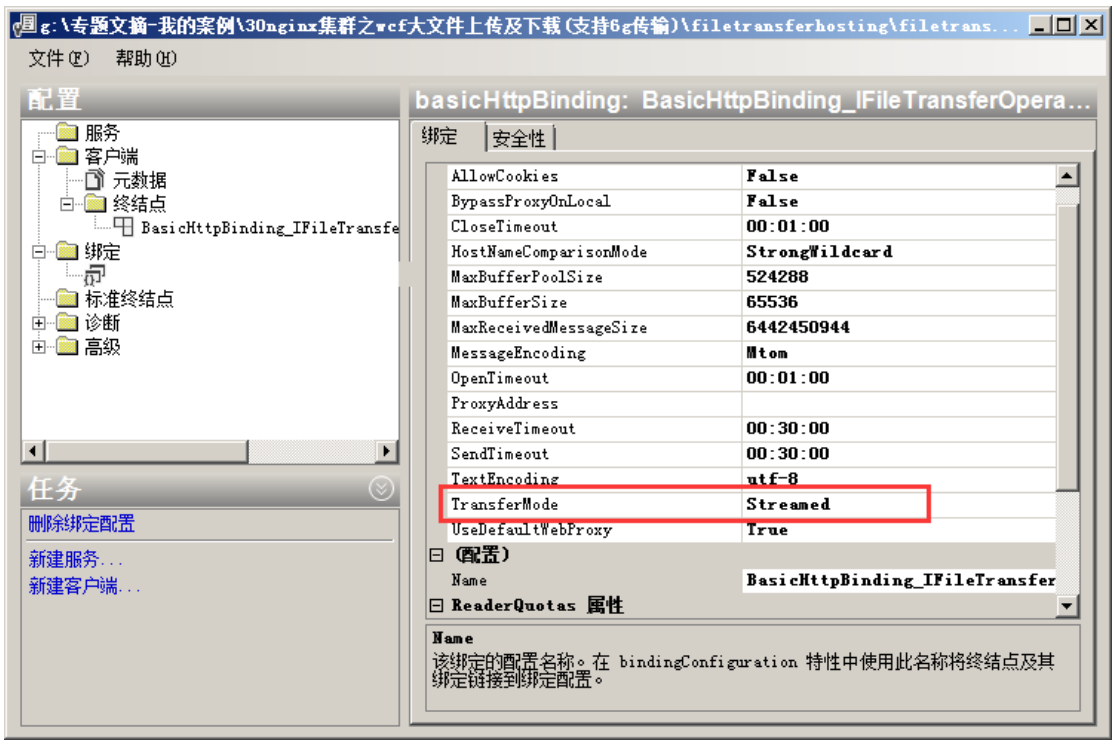
```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using FileTransferClient.FileTransferService;

namespace FileTransferClient
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("请输入文件完整路径：");
            string fullFilePath = Console.ReadLine().Trim();


            using (FileTransferOperationClient proxy=new FileTransferOperationClient())
            {
                DateTime datetime1 = DateTime.Now;
                byte[] buffer = System.IO.File.ReadAllBytes(fullFilePath);
                Stream streamBuffer= new MemoryStream(buffer);
                //上传文件
                proxy.UploadLoad(buffer.Length,  System.IO.Path.GetFileName(fullFilePath),
streamBuffer);
                OutPutDiffTime(DateTime.Now, datetime1, "上传成功");


                Console.WriteLine("开始下载文件：");
                DateTime datetime2 = DateTime.Now;
                string filename = System.IO.Path.GetFileName(fullFilePath);
                //下载文件
                Stream sourceStream = proxy.DownLoad(filename);
                if (sourceStream == null) { return; }
                if (!sourceStream.CanRead) { return; }
                //创建文件流，读取流中的数据生成文件
                using (FileStream fs = new
FileStream(Path.Combine(System.Environment.CurrentDirectory, filename), FileMode.Create,
FileAccess.Write, FileShare.None))
                {
                    const int bufferLength = 4096;
                    //数据缓冲区
                    byte[] myBuffer = new byte[bufferLength];
```

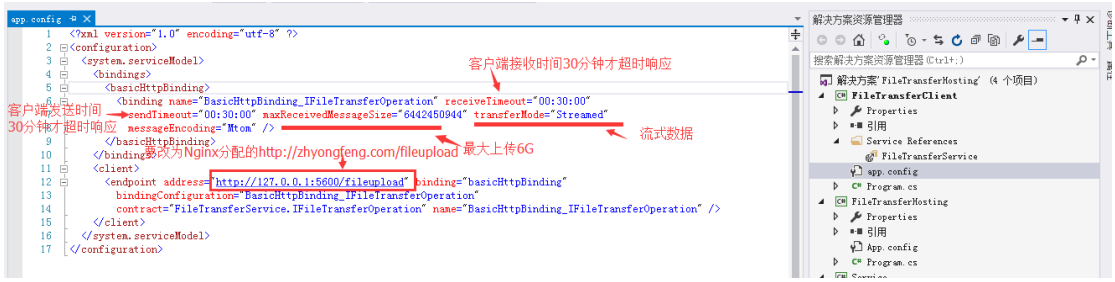
```
                    int count;
                    while ((count = sourceStream.Read(myBuffer, 0, bufferLength)) > 0)
                    {
                        fs.Write(myBuffer, 0, count);
                    }
                    fs.Close();
                    sourceStream.Close();
                }
                OutPutDiffTime(DateTime.Now, datetime2, "下载成功");
                Console.Read();
            }
        }
        public static void OutPutDiffTime(DateTime datetime2,DateTime datetime1,string
mesg)
        {
            TimeSpan ts = (datetime2 - datetime1);
            string dateDiff = ts.Hours.ToString() + "小时"
                    + ts.Minutes.ToString() + "分钟"
                    + ts.Seconds.ToString() + "秒";
            Console.WriteLine(String.Format("花费了{0},{1}", dateDiff, mesg));
        }
    }
}
```

客户端配置文件（注意这里客户端 transferMode=”Streamed”要采用流式数据进行处理，并且将 endpoint address 改为” http://zhyongfeng.com/fileupload”）：

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="BasicHttpBinding_IFileTransferOperation" receiveTimeout="00:30:00"
          sendTimeout="00:30:00"                maxReceivedMessageSize="6442450944"
transferMode="Streamed"
          messageEncoding="Mtom" />
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint address="http://zhyongfeng.com/fileupload" binding="basicHttpBinding"
        bindingConfiguration="BasicHttpBinding_IFileTransferOperation"
        contract="FileTransferService.IFileTransferOperation"
name="BasicHttpBinding_IFileTransferOperation" />
    </client>
  </system.serviceModel>
</configuration>
```

如下图所示：



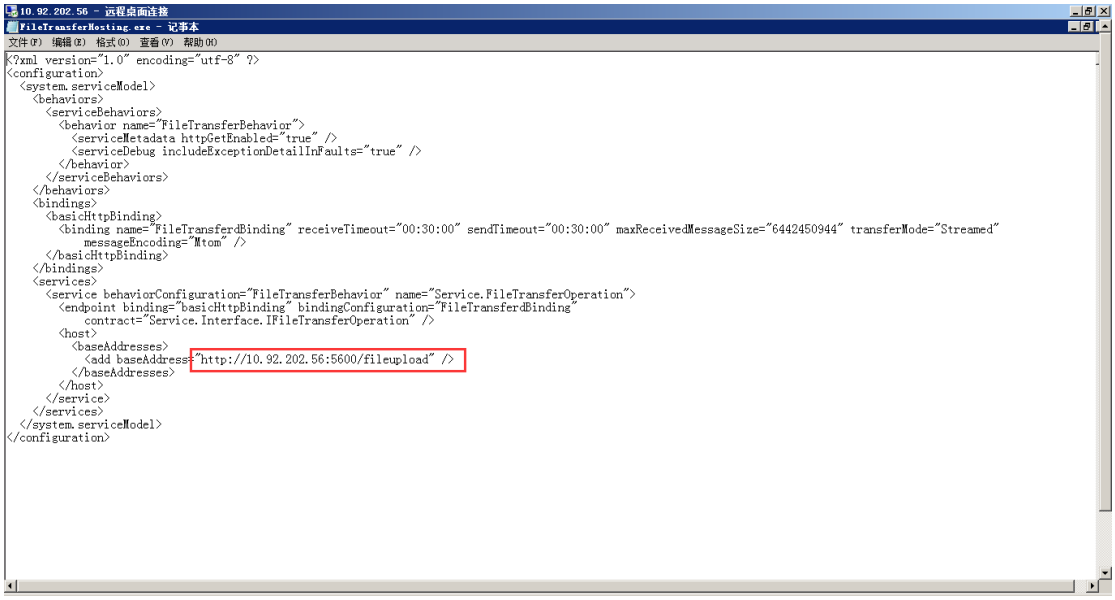


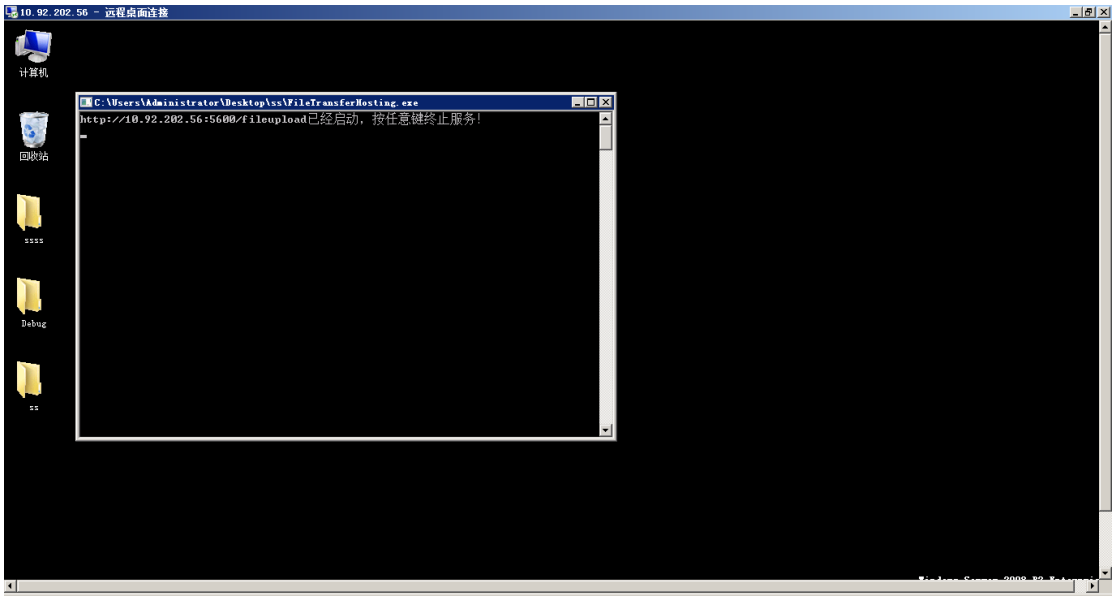
5 URL 保留项

详见：http://www.cnblogs.com/yongfeng/p/7851039.html

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：

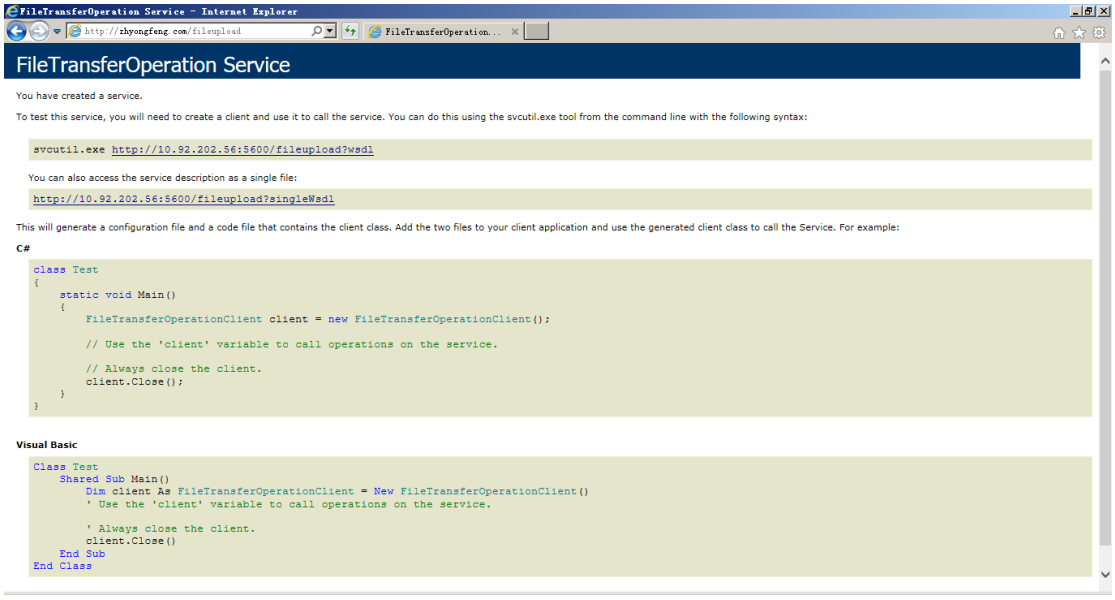
```
worker_processes    1;
events {
    worker_connections    1024;
}
http {
```

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

运行 CMD：

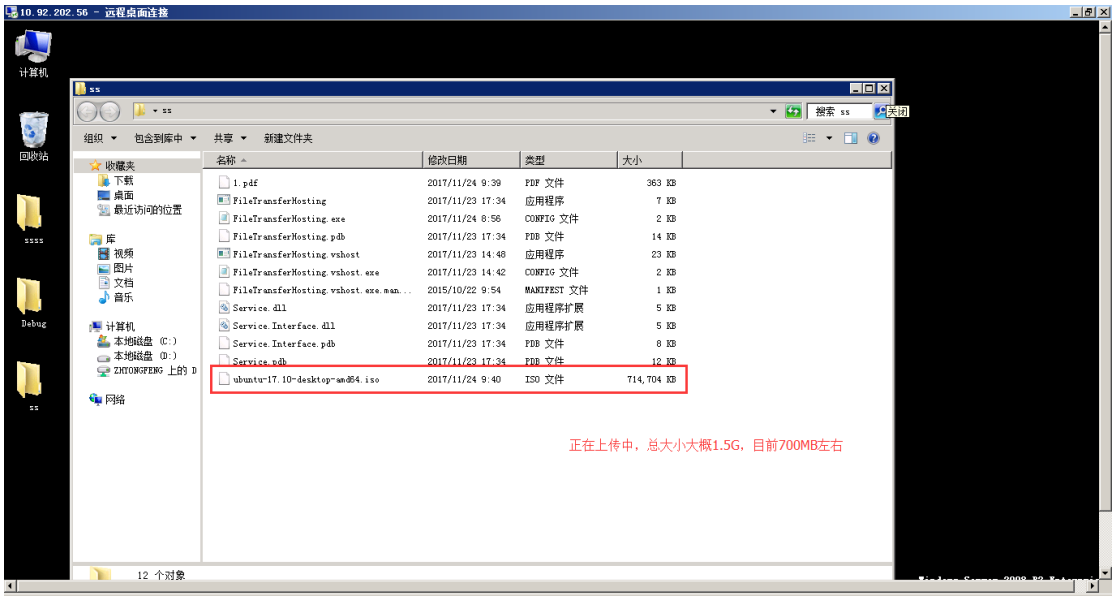
```
D:\DTLDownLoads\nginx-1.10.2>start nginx

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

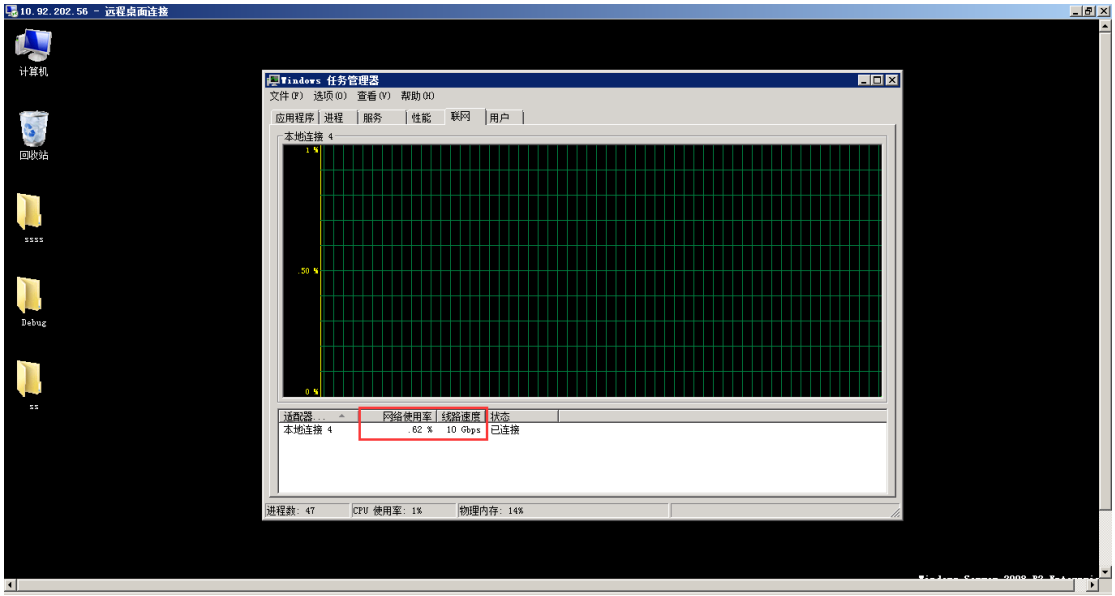
访问 WCF 服务端：http://zhyongfeng.com/fileupload，运行结果：

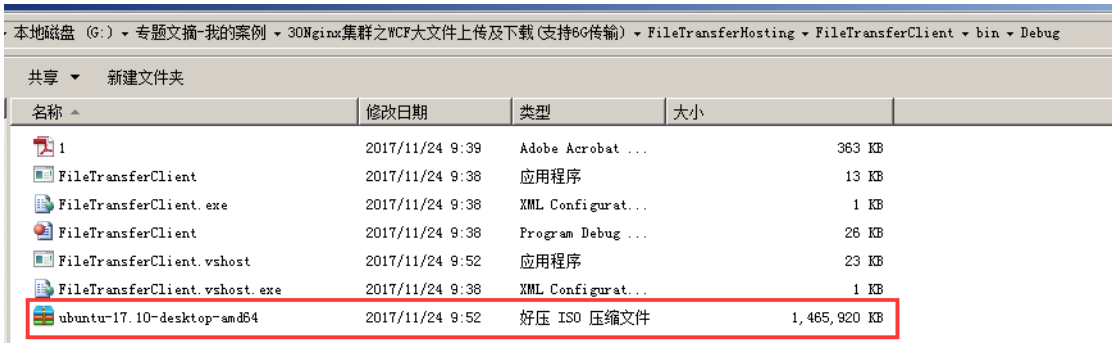


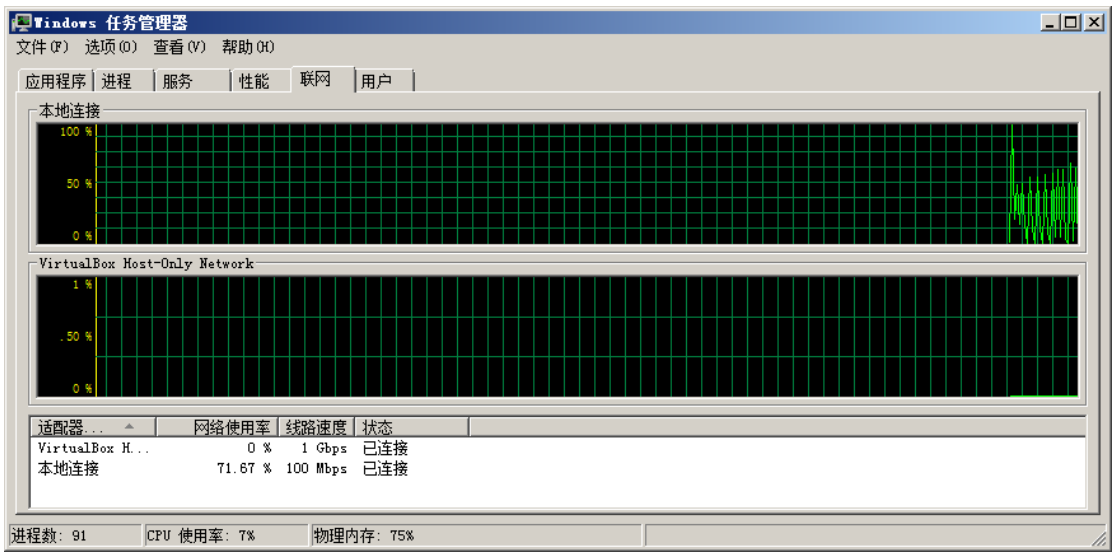
8 WCF 客户端程序的运行结果

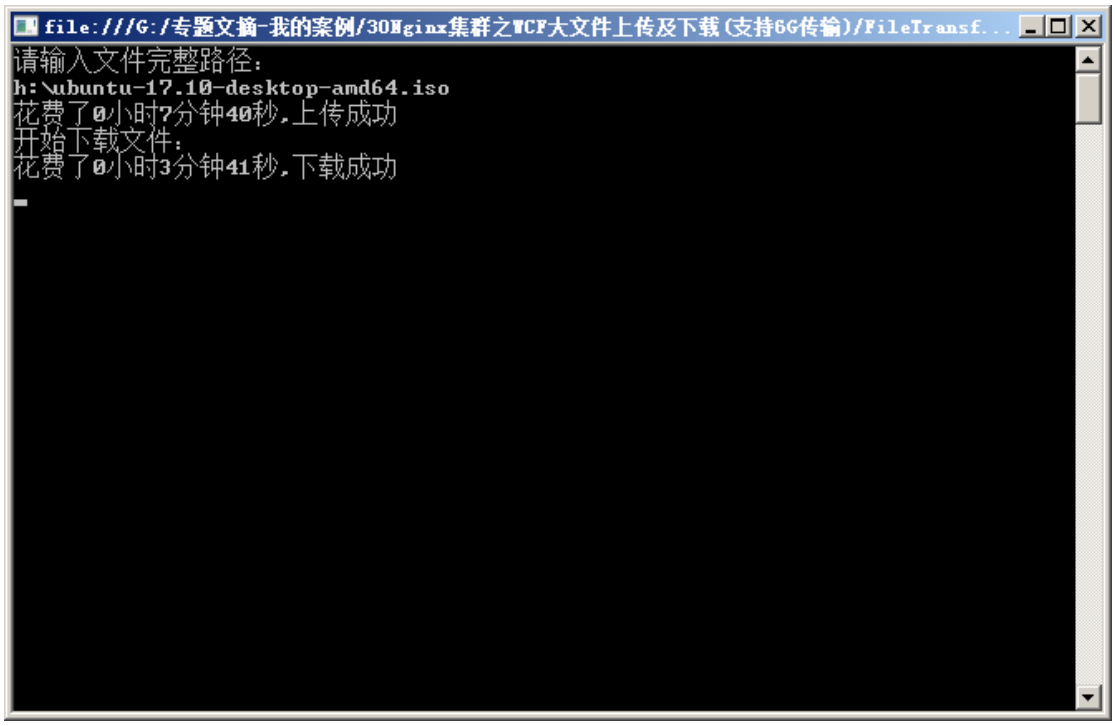
启动 WCF 客户端程序，直接上传 Linux 的 ubuntu 系统 iso 镜像，总大小为 1.5G 左右，上传时间局域网（服务器线路是 10Gpbs）花费 7 分钟左右，下载时间大概是 3 分钟。客户端服务器的网络状况，运行效果如下图：











9 总结

Nginx 的匹配原则能够有效的分配 URL，将流式数据分发给相应的服务处理，并且在局域网内能够支持较大的上传下载功能。通过 BasicHttpBinding 的相关配置，能够控制流式数据上传大小，同时支持流式数据的下载功能，达到 WCF 大文件上传及下载的效果。

当然，具体的应用场景，还是要结合数据大小而言论的，这里只是提供一个解决方案的参考。

例如一些处理视频文件每天都是 1TB，处理 PB 级的大数据文件，还是建议使用 hadoop 的 HDFS 实现比较好。

另一方面，数据大小达若干 MB 或几 KB 的发票、报表文件，这些文件主要在多而不在大，hadoop 的 MapReduce 显然有点大材小用了，可以采用“业务领域-年-月-日”的方式，建立自己一套数据结构存储方式。

源代码下载：

PDF 下载：
Nginx 集群之 WCF 大文件上传及下载(支持 6G 传输).pdf