进度显示的大文件上传组件的设计与实现

白 鹤[1,2],吕红亮[1,2],王劲林[2]
BAI He[1,2],LV Hong-liang[2],WANG Jin-lin[2]

1.中国科学院 研究生院,北京 100039
2.中国科学院 声学研究所 国家网络新媒体工程技术研究中心,北京 100190
1.Graduate University of Chinese Academy of Sciences,Beijing 100039,China
2.National Network New Media Engineering Research Center,Institute of Acoustics,Chinese Academy of Sciences,Beijing 100190,China
E-mail:web.baih@gmail.com

BAI He,LV Hong-liang,WANG Jin-lin.Design and realization of progress display on large file upload package.Computer Engineering and Applications,2009,45(5):91-94.

Abstract: File upload is a basic application on Internet,and the operation proposed in RFC1867 has some limitations on bad control ability and overtime for large file upload.An efficient large file upload package,UGiA-PHP-UPLOADER architecture based,is proposed,which can display progress.Its server listens high random port number for each upload connection,then parses received data according to protocol and generates temp status files while making use of hash mapping table to arrange these files;at the same time,the client requires data asynchronously with XMLHTTP to control progress bar.
Key words: XMLHTTP;file upload;progress display;large file;hash

摘 要:文件上传是互联网的一个基本应用,HTTP 协议中 RFC1867 实现的操作存在可控性差、大文件传输易超时等不足。基于 UGiA-PHP-UPLOADER 架构,提出了一种支持进度显示的大文件高效上传的方法。它在服务器端通过自定义高端端口接收上传内容,遵循协议解析并生成状态文件,并基于哈希映射算法对临时文件分配存储;客户端使用 XMLHTTP 技术异步请求服务端数据指导进度条。
关键词:XMLHTTP;文件上传;进度显示;大文件;哈希表
DOI:10.3778/j.issn.1002-8331.2009.05.026 **文章编号**:1002-8331(2009)05-0091-04 **文献标识码**:A **中图分类号**:TP393.09

1 引言

当今互联网迅猛发展的背景下,文件上传功能作为网络基本应用,其重要性日益俱增,同时也对其提出了更高的要求,但现存的各种文件上传技术从可控性、效率、体验各方面都存在不足。现今的上传技术按照架构可以分为两类:C/S 模式和 B/S 模式。

(1)C/S 模式上传技术,需要在客户端和服务器端安装支持软件,操作稍复杂,移植不便,通用性不强,但功能强大,可控性好。常见的有 ftp,广泛应用于互联网发展初期,以及一些版本管理软件。

(2)B/S 模式上传技术,一般基于 HTTP 协议中的上传规范(RFC1867[1]规范),其基本形式是 Form 表单,通用性强,定制部署方便,而且服务器端不需要提供 Web 服务外其他资源,但是其可控性差,无法获得即时状态;同时表单上传会受到 Web 服务器在连接时间和文件大小方面的限制,对大文件支持不好。基本形式的扩展包括一些 ActiveX 或 Applet 实现的第三方组件,一定程度上克服了相应不足,但是或者会因为安全性因素被浏览器限制使用,或者需要安装额外的环境组件(如 JVM 等)。

B/S 模式技术组成了现有网络上传技术的主流应用,但是存在前述种种不足。本文通过对 RFC1867 规范重新解析和封装,实现一个无第三方组件、可显示进度、支持大文件上传的工程方法:用 C 语言实现文件接收的服务器端,其监听预定义随机高端端口,按照 RFC1867 规范重新解析接收内容,记录并基于哈希映射算法分配实时存储状态信息;用 PHP 实现了 Web 响应服务器端,读取状态信息响应请求;用 JavaScript 构建基于浏览器的客户端,发送请求接受结果,调用 DOM 接口控制实时进度显示;应用 XMLHTTP[2]负责上传过程异步通信。此方法接手了 Web Server 对表单上传的控制,摆脱了大文件上传会出现的时间限制,加快了处理速度,并且可以实时进度显示。

2 B/S 模式文件上传技术现状

HTTP 是一种基于请求/响应模式的协议,一个客户机与服务器建立连接后,发送一个请求给服务器,请求包括:请求的方法、URI(Uniform Resource Identifier)、协议版本号。随后还有

基金项目:国家高技术研究发展计划(863)(the National High-Tech Research and Development Plan of China under Grant No.2008AA01A307)。
作者简介:白鹤(1982-),男,博士研究生,主要研究领域为信息挖掘等;吕红亮(1983-),男,博士研究生,主要研究领域为宽带通信等;王劲林(1964-),男,研究员,博士生导师,主要研究领域为宽带通信、移动多媒体等。
收稿日期:2008-08-07 **修回日期**:2008-10-21

MIME(Multipurpose InternetMail Extension protocol)信息:包括请求修饰符、客户机信息和可能的内容。服务器接到请求后,给予相应的响应信息,其格式为一个状态行,包括信息的协议版本号、一个成功或错误的代码,后边是 MIME 信息包括服务器信息、实体信息和可能的内容。而基于 FORM 表单的文件上传,文件的信息和内容正是封装在 MIME 信息内发送到服务器端的。

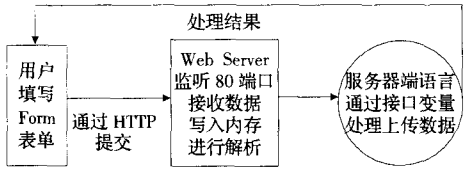

图 1 RFC1867 协议流程

基于浏览器的文件上传有三种机制:RFC1867、PUT 和 WebDav。只有遵循 RFC1867 规范的机制可以使用服务器端脚本语言编程处理,获得广泛应用,主流的浏览器、Web Server 和 Web 服务器端语言均已支持此协议。其处理步骤如下:

(1)用户在浏览器端填写特定类型(multipart/form-data)的 Form 表单,并提交;

(2)服务器监听 Http 服务端口(一般 80),接收表单数据;

(3)Web Server 按照规范解析表单,使用服务器端程序(PHP、JSP、ASP)的接口变量获得解析后的内容进行处理;

(4)将执行结果返回给用户。

其通用性强,定制部署方便,服务器端不需要提供 Web 服务外的其他资源,但是其可控性差,无法获得即时状态,又因为受限于 Web Server 对大文件上传支持不好。这就导致了封装 RFC1867 协议的第三方组件大量出现:

(1)JAVA 实现的组件

常用的有 Commons-fileupload[3]和 smartupload:前者缺点是把上传数据流全部写入内存,导致内存占用过多,并在上传大文件时会受到 Web 服务器限制;后者是 Jakarta 一个项目中的组件,没有进度控制,用户体验不好。文献[4]的上传组件支持进度显示,但是基于 JAVA 实现的接收文件处理服务需要经过虚拟机层的编译,效率不高。另外存在 Java 实现的 Applet 小程序,可以嵌入网页,绕开 RFC1867 实现上传,其本质是一个小客户端,并且需要安装额外的 JVM 插件。

(2)ASP 实现的组件

常用的有 SlickUpload、SA FileUp,其性能优越,但是需要付费,源码没有公开,就不能根据具体需求适应更改更新,而且使用 ASP 语言只能适用于 Windows IIS 服务器。

(3)Activex 上传组件

使用 ASP 调用组件,可以完成较丰富的上传功能,但是因为安全性因素,Activex 控件在用户端会被浏览器控制选项禁用。提出的文件上传解决方案,基于 UPU(UGiA-PHP-UPLOADER[5])组件的系统架构,使用 C 语言实现核心的文件接收服务器端,同时应用基于 Hash 表的算法改进了上传过程中的文件管理。它支持进度显示和混合表单上传,不需要插件,可以适用于任何浏览器客户端,满足了不同平台的可移植性。

3 文件上传解决方案

进度控制的大文件上传解决方案是基于 RFC1867 规范实现的,有三个要点:本地表单提交后的控制,服务器端数据流接收、解析和日志保存,客户端和服务器端的异步信息交互。本方案不依赖第三方组件,分别使用 Javascript、PHP 和 C 语言构建了客户控制端、服务响应端和数据接收处理端三个主要模块。系统架构图如图 2 所示。

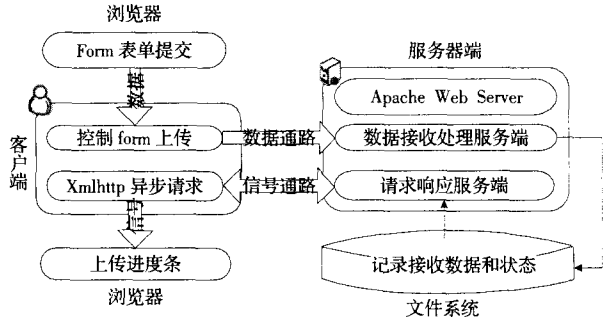

图 2 系统架构图

浏览器完成表单提交和进度条的显示;客户端负责对表单处理的控制和定时请求服务端的状态日志以得到信息操作进度条;数据接收处理端和服务响应端两个模块在服务器端,分别负责接收、解析数据和响应客户端异步请求;文件系统主要负责存储文件和状态日志。

3.1 客户端控制

客户端控制主要通过 Javascript 脚本语言来实现,所有主流浏览器都支持对它的编译。客户端与服务器通信是通过 XMLHTTP 异步技术,它是传送 XML 格式数据的超文本传输协议;其实它的数据传输形式非常灵活:它上传的指令和下达的结果可以是 XML 格式数据,也可以是字符串、流,或者是一个无符号整数数组等;通过此协议可以方便地在异构平台之间进行数据交换。目前,绝大多数浏览器都增加了对 XMLHTTP 的支持,IE 中使用 ActiveXObject 方式创建 XMLHTTP 对象,其他浏览器,如 Firefox,Opera 等通过 window.XMLHTTPRequest 来创建 XMLHTTP 对象。调用过程很简单,一般有以下 5 个步骤:

(1)创建 XMLHTTP 对象;

(2)打开与服务端的连接,同时定义指令发送方式,服务网页(URL)和请求权限等;

(3)发送指令;

(4)等待并接收服务端返回的处理结果;

(5)释放 XMLHTTP 对象。

由此本组件无需插件可以适应各种浏览器环境。

用户填写 Form 表单点击提交,触发了事件反应函数 Onsubmit,函数进行文件合法性判断后,会打开新页面显示进度条,重要的是给表单返回"false"信号阻断浏览器上传表单,相当于将 Form 控制权柄交给客户端,其由新打开进度条显示页面内嵌的 Javascript 代码构建,完成如下功能:

(1)数据发送

客户端启动定时器,并初始化 XMLHTTP 的实例,设置 GET 方法定时发送异步请求到服务器请求响应端轮询状态,得到端口号信息代表已做好接收数据准备;得到父页面表单对象,设置"action"属性为"http://ip:port",即设置非 80 新的上传端口,启动"submit",开始表单数据和文件的上传。同时初始化进度条页面:设置抬头信息"开始上传"。

(2)进度条处理

客户端发送数据后,服务器接收数据并监督记录状态,客

户端定时请求获得数据,通过 DOM[6]接口处理进度条的即时状态显示。文档对象模型 DOM(Document Object Model),是一种与浏览器、平台、语言无关的接口,用于开发者访问页面中其他的标准组件。主要应用于 getElementById 接口函数,输入组件 id 获得其对象。

进度条会随着接收状态顺次显示服务器端口信息、文件名字长度信息、上传中进度信息、上传结束信息。其中进度条显示的处理需要根据返回的当前接收数据长度和文件总长度,计算比值,倍乘以进度条总长度,得到进度即时长度,然后对获得的控件对象长度进行赋值。在这个过程中,还要统计上传速度和剩余时间。前者的计算使用当前接收数据长度与消耗时间的比值,消耗时间在客户端用定时器累加一个全局变量得到;后者的计算,使用剩余数据长度与之前得到的上传速度的比值。速度和时间的数值换算成文字表述再显示。比如时间统计以秒为单位,判断其数值若大于 3 600,就整除 3 600 获得小时数,此步余数整除 60 得到分钟数,余数为秒数。

(3)发送结束

客户端接收到的传送完成信号是一个容纳 Form 表单的数组,其重新组织了表单的数据项,关键是将 file 控件替换成值为接收文件存放位置的 text 控件,然后通过 http 提交表单,客户端工作结束。另外存在一种非正常中断,用户在上传中间点击取消或者关闭进度页面同样导致客户端生命周期结束。

3.2　服务器端处理

服务器端是构建于 Web Server 内负责上传文件处理的组件,主要包括数据处理和请求响应两个模块,前者接收分析数据,后者读取日志响应客户端请求。

(1)数据处理服务端

服务端接收文件,首先初始化一个高端随机端口,端口信息传抵客户端后就实例化 Socket 对象、监听此端口,这样就脱离了 Web 服务器对表单上传的控制,同时摆脱了其对文件长度和传输时间的限制。遵照 HTTP 协议编写 200 的响应头,回复给 Post 命令请求此端口的客户端,随后按照一定字节长度循环读取 Socket 数据流,压入一定长度的字符队列,在循环次数是队列长度的整数倍次时将队列内容写入磁盘文件,并按照 RFC1867 规范解析队列中内容,获得 HTTP 头里面的长度、类型信息和 Form 表单中除了 file 控件其余的内容,写入状态日志。这个模块中,分段读取长度一般设置为 1 K 字节,防止一次读取占用内存过多;采用即时解析处理的策略,首先可以满足客户端定时获取长度和状态信息的需求,另外避免了传输结束再处理时需要对整个文件读取造成的时间和资源浪费;定长队列是即时解析的数据对象,不直接使用新获取数据是因为防止成块结构信息被切断造成不能识别;写入磁盘文件过程采用了"滞后合并"策略,相比读取后即时写入,本方法在不增加内存占用情况下减少了与磁盘 I/O 交互,提高整体速度。

RFC1867 规范是解析的理论依据,有如下几个关键结构点:

①头文件里面的"Content-Type",以分号相隔赋值的第二部分,"boundary"的取值是 27 个"-"和 13 个随机字母数字相连的字符串,这就是表单各部分的分隔符;

②每个分隔符换行之后就是表单各控件的输入具体内容,分号分开的各部分有"name"、"value"、"Content-Disposition"等值;如果是文件控件,还会有"filename"、"Content-Type"分别代表在客户电脑存储路径和上传文件类型;

③数据流结束符是分界符链上两个"-",换行后是以"ok"结束。

根据以上结构点用 C 语言构造正则表达式对于队列内容可以得到有效快速的解析。

(2)服务响应服务端

此模块按照客户端请求阶段读取状态对应临时文件,如果获取成功就组织数据,回应请求。成功的文件上传中,此信息通路共有 5 次握手,具体见图 3。

阶段 1 服务端读取 srv 文件,获得服务器和端口信息;客户段得到后提交表单内容到对应端口,初始化进度窗口。

阶段 2 服务端读取 con 文件,获得上传混合数据总长度;客户端设置进度条。

阶段 3 服务端读取 inf 文件,获得上传文件信息,包括文件名和文件类型;客户端设置进度条。

阶段 4 服务端计算 dat 文件长度;客户端以即时长度计算剩余时间等。

阶段 5 服务端重组 form 表单信息,客户端获得 form 后上传正常的 80 端口,将控制权柄交给 RFC1867 处理。

握手结束后,会删掉临时状态文件,请求响应端任务结束。

3.3　文件系统存储

本模块负责存储接收的数据和状态临时文件。每个上传对象的临时文件共 5 个,文件类型包括:srv、con、inf、dat、frm,分别存储服务器端口信息、上传数据长度信息、正在上传文件信息、原始数据、新构建表单。文件按照既定流程(见 3.2 节)生成

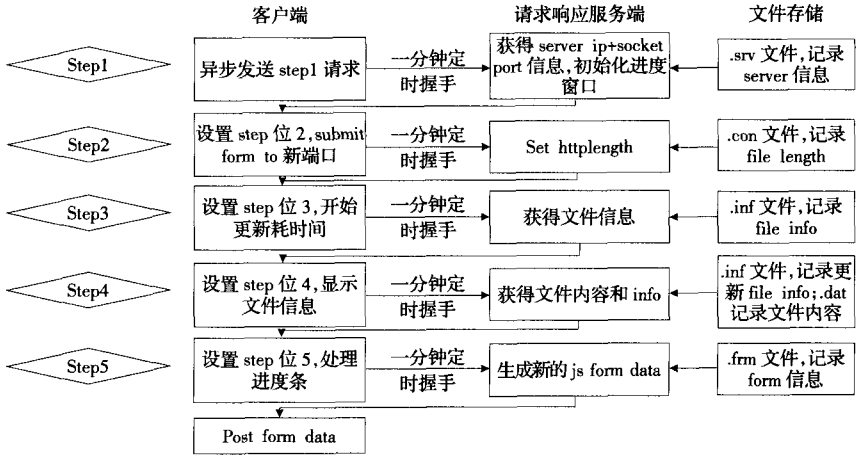

图 3　信息通路 5 次握手

读取,文件的存储分配策略会影响到存取效率,在上传并发量大的情况下,文件存储成为了系统瓶颈。

首先要解决文件命名的问题,如果选择原文件名,针对热门对象就会出现名字重复而带来存储上的冲突。为避免这种情况,状态文件起名采用32位随机数,计算方法是:即时时间叠加随机数,随后用md5运算。接着就是考虑分配存储的策略,最简单的莫过于所有状态数据文件存储在同一个文件夹内,弊端显而易见:组件在频繁进行磁盘I/O操作时,文件夹内文件数量的增多就会极大影响读写速度。所以需要根据上传负载确定文件夹数量,然后遵循一定策略分配。上传不具有对于时间的线性关系,所以不能选择基于时间的策略;考虑到文件名是随机数的特征,采用基于文件名的分配方法。对于数量大的随机字符串的均匀分配,哈希是一种高效的算法,只用$\Theta(1)$时间效率就可以将复杂key值独立地映射到对应区(slot)。哈希函数是算法的核心,常用的主要有两类:分割(Division)和倍乘(Multiplication)方法。倍乘函数原型如下:

$$hash(k)=(Ak \bmod 2^{\omega})>>(\omega-r)$$

此方法运算主要集中在取2的幂和移位,计算简单,但是对乘数、除数和输入位数要求严格。分割函数原型:

$$hash(k)=k \bmod m$$

此方法原理简单,m取值不受输入的影响,为了保证输入的每一位都起作用,只要求m不取2的幂。分析模块需求,要将数量庞大的随机字符串作为Key值(状态文件名)均衡分配到有限Barrel(文件夹)中,选取Division类哈希函数比较合适。具体函数设计如下:

$$hash(key)=\sum_{i=1}^{key.length} Ascii(key(i)) \bmod P$$

$$P\neq 2^{\omega}(\omega\in(0,1,2\cdots))$$

对文件名中每一位字符的ASCII码求和,其值作为分割的输入;P是根据负载设置的文件夹个数,限制是不能取2的幂。函数中主要是加法和取余运算,计算效率较高;同时随机文件名中每个字符都会影响到hash结果,并且每个字符的影响是近似均等的,这就保证了函数性能上的对输入Key值的均衡分配。第4章的仿真可以验证其分配效果。

4 仿真和讨论

分别在局域网和广域网内搭建了上传组件服务器。局域网内实验可以控制网络和服务器状态,主要测试前后组件性能不同和哈希映射分配策略的效果;广域网内实验效果会受到即时带宽和服务器性能状况影响,主要通过与成熟站点的文件上传模块的比较,总结功能特性和上传速度之间的差异。

在局域网内,服务器和客户端机器之间通过100 Mb/s交换机相连,服务端是1 G内存、P4.2.8 G的CPU、安装FreeBSD系统,客户端使用IE7.0浏览器。针对UPU组件和新组件,各自上传同样的10个音乐和视频文件,最小的文件大小是2.4 M,最大的700 M,记录下平均速度。

(1)对两组数据各自计算算术平均,得到UPU的平均速度2.72 MB/s;新组件是3.2 MB/s。可见经过改进的新组件可以平均提高近18%的处理效率。单独考虑700 M大文件的上传情况,其使用新组件上传速度相比UPU提高了30%,这说明了文件越大,处理效率提高越明显。究其原因在于小文件在有限几次循环读取和解析中就可以上传结束;而大文件的循环读取次数可达10万的数量级,UPU在关键处理上耗费性能累加会加剧负担,越发延缓处理,所以新上传组件针对大文件提升更多性能。

(2)实验中新组件的文件系统设置了三个文件夹(即Hash函数中P=3),在运行结束时不删除临时状态文件,依此调查哈希映射的分配效果,结果是两个文件夹各自存了4个上传对象的日志,剩余一个是两个;在只有10个样本的实验中,此分配效果已较均衡,Hash函数可以达到预期效果。

在广域网内,组件的服务器在电信机房,与机柜内机器共享3 MB带宽,配置类似于局域网内测试机器,安装新上传组件;客户端机器处于科技网内。测试方案中选择的比较对象是土豆网,这是一个热门视频网站,上传是它的重要模块,开发比较成熟。

(1)功能特性对比见表1:

表1 新组件与土豆上传模块功能对比

	上传组件	土豆上传模块
支持进度条	是	是
支持大文件	理论无上限	200 MB
无需插件	是	是
可移植性	适应各种平台	分析使用PhP,此性能也较好

可以看出就功能丰富来说差别不大,只是在大文件支持上,新组件要更优越一点。

(2)上传速度对比。此项对比要真实反应组件性能,就要尽可能去除网络和服务器硬件差距带来的影响,理想情况是带宽充足、服务器资源闲置较多,所以选择早上7点钟。上传文件是168 MB视频文件,新组件平均速度472 KB/s;土豆的则是409 KB/s,可见本组件上传速度达到了较高水平。

(3)稳定性对比。选择早上7点和晚上8点各上传一组5个文件,对比双方早上组的测试成功率都是100%,但是晚上组新组件最终上传成功的只有3个文件,土豆是5个,可看出在带宽和服务器资源不足的情况下组件的稳定性不够,这是需要进一步完善的地方。

5 结论

基于HTTP的文件上传操作部署方便,但是存在不支持进度条和对大文件支持不好等诸多不足,基于UPU组件架构,打开高端端口接收数据,遵循协议解析并生成状态文件,又基于哈希映射算法对临时文件分配存储,客户端使用XMLHTTP[4]技术经过与服务端五次握手请求数据指导进度条,实现基于RFC1867支持进度条和大文件的文件上传组件,经过实验本组件性能已达到较高水平。

参考文献:

[1] HTTP协议上传文件规范[S/OL].http://www.ietf.org/rfc/rfc1867.txt.

[2] W3C DOM规范[S/OL].http://www.w3.org/DOM/.

[3] 王文龙,王武魁.利用Java语言实现文件上传功能[J].微计算机信息,2007(11):169-171.

[4] 段旭光,李岚,董立岩,等.基于HTTP文件上传进度显示组件的设计与实现[J].吉林大学学报:信息科学版,2006(24):89-93.

[5] UGiA-PHP-UPLOADER开源项目[EB/OL].http://sourceforge.net/projects/upu.

[6] W3C DOM规范[S/OL].http://www.w3.org/DOM/.