计算机科学1999Vol. 26№. 3

有态分布式 HTTP 的实现

An Implementation of Stateful and Distributed HTTP

徐瑞斌

（中山大学计算机科学系　广州510275）

Abstract *Standard HTTP can not meet the needs of current powerful WWW applications because of its lack of state management and distributedness. This paper specifies ways to create stateful sessions with HTTP request and response, and realize effective cooperation of multiple WWW servers, ie. implement stateful and distributed HTTP.*

Keywords WWW, HTTP, Web Farm, Distributed Application, Cookie

1. 前言

WWW(World Wide Web)是 Internet 上最主要的服务，在 WWW 中，Internet 用户通过 WWW 客户(浏览器)获取分布在全世界各地的 WWW 服务器的超媒体信息，它们之间传输信息的标准协议是 HTTP(HyperText Transfer Protocol，超文本传输协议)。HTTP 采用请求/响应方式进行通讯，其运作的基本过程是：

(1)客户端与服务器建立连接；

(2)客户端向服务器提出请求；

(3)服务器响应客户端的请求；

(4)客户端与服务器断开连接。

以上四个过程构成 HTTP 的一次事务，所以 HTTP 是无状态的，并且在这个结构当中，WWW 服务器之间没有通讯，信息的传递只是在服务器和客户端之间进行。当客户端需要从多个服务器获取信息时，由客户端自己负责识别所需要访问的服务器。由于 HTTP 具有通用、无状态以及面向对象的特点，所以非常适合静态的协作化的超媒体系统。但在 WWW 上实现复杂交互式操作、动态查询等客户/服务器(C/S)模式的网络应用，特别是当今流行的 WebDB 应用方面，HTTP 就显得力不从心了。HTTP 欠缺的是：

(1)有态性。C/S 系统中客户和服务器之间的协议是有状态的。状态是存储在服务器端与某个客户之间交互的有关数据的总和，存放关于通讯历史的状态信息需要依赖于有态协议的支持，即消息含义依赖于前面消息的协议。WWW 服务器通常总处于一种状态，对 WWW 客户端的反应不依赖于以前请求和反应消息的历史。HTTP 是无状态的，这种无态协议只适合简单的应用。

(2)分布性。在 HTTP 通讯方式下，WWW 服务器在逻辑上是孤立的结点，它们之间通常没有通讯联系，多个服务器的访问全由客户端识别处理，这样就很难实现复杂的分布式应用。然而，分布式应用要求多个服务器以某种规范的方式进行通讯，而这种通讯对客户端是透明的，以达到均衡计算，共享资源(包括 CPU、存储空间等)，减少信息冗余和加快交互效率的目的。

要解决 HTTP 缺乏有态性和分布性之先天不足，我们将寻求 HTTP 本身和以外的一些关键技术。

2. 实现有态 HTTP

在 WWW 中，一个 WWW 服务器可同时与多个 WWW 客户进行对话，每一个对话都是由多次 HTTP 的请求/响应所组成。然而 WWW 服务器是无状态的，所以它并不能识别每一次请求是属于哪个对话，当然也就不能对提出请求的客户作出正确的响应。因此，解决问题的关键是：WWW 服务器每次收到的请求信息中必须含有状态的有关信息。

WWW 服务器为每一个对话设置一个对话标识(SessionID)，并以对话标识为主键(保证唯一性)，利用文件系统或数据库存储与该对话对应的客户交互的状态信息；而 WWW 客户的每一次请求都

必须携带它所属对话的标识。这样，WWW 服务器收到客户请求时，就从请求信息中提取出对话标识，再根据该对话的状态信息，产生相应的响应传送给客户。

以下是一个 WWW 客户与 WWW 服务器对话的全过程：

(1)WWW 客户向 WWW 服务器发出第一次请求(不含对话标识)；

(2)服务器收到请求后，辨别出请求不含对话标识，则生成一个具有唯一性的对话标识(可利用客户的 IP 地址、请求的时刻等作为参数)，并在文件系统或数据库生成相应的记录，转(4)；

(3)客户向服务器发出请求(含对话标识)；

(4)服务器根据请求信息中的输入参数和该对话的状态信息，执行相应的动作，生成相应的响应(含对话标识)返回给客户，并同时修改该对话的状态信息，以供响应客户下一次请求时使用。转(3)。

一个对话的开始是由 WWW 客户发起的，而结束则由 WWW 服务器来决定，因为服务器每一次响应客户请求后都与客户断开连接，所以服务器并不能预知客户下一次请求的时刻。若客户退出，服务器是不知道的。因此，服务器可对每一个对话设置一个时间限制值(Timeout)，若某次请求后超过这个时间限制值还没有请求，服务器认为客户已退出，就自动结束这个对话。在传送对话标识的具体实现上有三种方法：

(1)利用请求 URL 中的参数，如

http://server/path?SessionID=....

(2)利用 HTML 文件中表单(Form)的 Hidden 输入元素，如：

```
.....
<FORM>
    .....
    <INPUT TYPE = "hidden" NAME = "SessionID"
VALUE="    ">
    .....
</FORM>
.....
```

以上两种方法存在两点不足之处：(i)若对话中存在静态页面响应时，则实现起来较为烦琐。(ii)安全性问题。以上两种方法传送的对话标识，都可以被 Internet 用户轻而易举地从浏览器获得，因此网络黑客可以修改服务器生成的对话标识(如改成另个一个对话的标识)，在向服务器发出请求，使服务器作出错误的响应，并有可能破坏服务器的数据。鉴于以上原因，我们可以用第三种方法：

(3)利用 Cookie。服务器收到客户的第一次请求后，产生一个对话标识，然后在 HTTP 响应头里用：

Set-Cookie:SessionID=

之后客户每次向服务器发出请求时，在 HTTP 请求头里都会有：

Cookie: SessionID= 、

这样实现显然比前两种方法简便。不过需注意的是，利用 Cookie 来传送对话标识的安全性并不是绝对的，只是比前两种方法效率都高。

3. 实现分布式 HTTP

WWW 上的分布式应用的特征是一组 WWW 服务器以某种对 WWW 客户透明的方式互相协作，共享资源。这里需要解决两个关键问题：(1)WWW 服务器通过什么方式进行通讯；(2)如何控制 WWW 客户对 WWW 服务器组里的各个服务器的访问。

对于第一个问题，若 WWW 服务器组里的服务器的物理位置都比较靠近，就可以把它们连接成一个逻辑星型局域网，中心结点不是 WWW 服务器，而是一个文件服务器或数据库服务器，各 WWW 服务器通过中心结点进行通讯(共享目录或 SQL 操作)。局域网采用比 TCP/IP 更加有效率的协议(如 NetBEUI)。如果 WWW 服务器组里的各服务器相距较远，则它们之间的协作方式与上述大体相同，只是采用的网络协议是 Internet 上的 TCP/IP，效率差一点而已。对于第二个问题，要分两种情况：

(1)WWW 服务器组里的每一个服务器都具有相同的功能。这些服务器通常是同一个域下的不同主机，这种服务器组被称为 Web 农场(Web Farm)，如 www1. domain，www2. domain，……。WWW 客户访问其中任何一个服务器都得到相同的服务，所以逻辑上可把此服务器组看成是一个 WWW 服务器，设为www. domain。此时，为 WWW 客户分配要访问的服务器有两种方法：

(i)随机分配。www. domain是一个虚拟主机，即不指向具体的主机，它所对应的 IP 地址是可变的，可以是 www1. domain，www2. domain，……之中任意一个服务器的 IP 地址(负责解释该域域名的系统称为动态域名系统)。WWW 客户发出对www. domain的请求前，先向动态域名系统发出对www. domain的解释请求，则动态域名服务器随机选择服务器组里任意一个服务器的 IP 地址作为响应，这样就使 WWW 客户去访问该 IP 地址所对应

的 WWW 服务器。

(ii)智能分配，www.domain是一个具体的主机，有固定的 IP 地址。在服务器组里的每一个服务器中都运行一个代理(Agent)进程，负责实时收集该服务器的 CPU 使用率、正在进行的对话的数量等资源使用情况，并不断通过网络向www.domain报告。当 WWW 客户向www.domain发出请求时，www.domain根据均衡原则和各服务器的资源使用情况，选出一个合适的服务器(如 www2.domain)，然后产生响应码为302(Object Moved)的 HTTP 响应：

```
HTTP/1.0 302 Object moved
Location: www2.domain
……
```

WWW 客户(浏览器)收到此响应后，会自动向www2.domain发出请求，这就是 HTTP 响应的重定向功能。注意在此过程中，实际上还是要 WWW 客户识别要访问的服务器。

(2)WWW 服务器组里的服务器分别执行不同的功能。在这种情况下，一个 WWW 客户与服务器组只存在一个对话，也就是说，对于一个 WWW 客户，服务器组里的所有服务器共享一份该客户的状态信息。对话的状态信息存储在服务器组的中心结点里。一个 WWW 客户切换访问服务器里的服务器时，对话的标识保持不变。现在的关键问题是如何在服务器之间传递对话标识。

一种方法是可以利用 URL 参数进行传递，但会出现像在实现有态 HTTP 时的不足之处。常用的方法还是利用 cookie 来传递，这里有两种情况：

(i)服务器组的服务器是同一个域(设为 domain)下的不同主机，也就是构成一个 Web 农场，则客户访问的第一个服务器(由它生成对话标识)发出的 cookie 应设置它的 Domain 属性，即

Set-cookie：SessionID=……Domain=domain

之后，WWW 客户向服务器组里的每一个服务器发出请求时，因为都是同一个域，所以 HTTP 请求头都会携带该 cookie 的信息，也就是携带对话标识的信息。

(ii)服务器组里的服务器的域不同。这就需要用 URL 参数来配合实现对话标识的传递。设服务器组里只有两个服务器，分别是www.domain1和www.domain2。WWW 客户首先访问www.domain1，取得一个对话标识。之后，WWW 客户切换访问www.domain2，则记录对话标识的 cookie(属于 domain1)不会传递过去，这时www.domain2就需要向 www.domain1 询问对话标识，www.domain2把当前 WWW 客户访问自己的 URL 当成一个参数，设为 return-address，然后发出如下响应使 WWW 客户重定向去访问www.domain1的一特定文件，设为 response：

```
HTTP/1.0 302 Object moved
Location: www.domain1/response? return-address=……
……
```

此时，记录对话标识的 cookie 就会传递给 response，而 response 则发出如下响应使 WWW 客户再重定向去访问www.domain1的 return-address，并向 WWW 客户设置一个新 cookie，它的 Domain 属性是domain2：

```
HTTP/1.0 302 Object moved
Location: www.domain2/return-address
Set-cookie: SessionID=……Domain=domain2
……
```

这样，www.domain2就可以接收到对话标识。

上面是一种特殊情况，并且要求 WWW 客户先访问www.domain1，再访问www.domain2。若服务器组里有多个服务器，则需要把服务器组里的服务器连成一个询问圈，每个服务器收到客户请求时，若请求信息中不含对话标识，则需要按上面所说的方法顺着询问圈来询问对话标识。如果询问请求回到了自己本身，就建立一个新的对话标识。

结束语 遵循上述实现有态分布式 HTTP 的方法，笔者首推 ASP(Active Server Page)作为一种实现工具，利用它可以产生和运行动态的、交互的、高性能的 Web 服务应用程序，有兴趣的读者可以参阅其他资料。

参 考 文 献

1 赵洪彪、周立柱. 数据库系统与 WWW 的集成途径. 计算机科学，1997(6)

2 Jamsa K，等. WEB 程序设计教程. 电子工业出版社，1997

3 Kristol D，Montulli L. RFC2109，HTTP State Management Mechanism，February 1997

4 Available at：http.//home.netscape.com/newsref/std/cookie spec.html