# 嵌入式 Web 服务器的设计与研究

任 斌

(江苏食品职业技术学院,江苏 淮安 223003)

**摘 要**：随着计算机技术和网络技术的快速发展,以嵌入式设备为主的监控系统、信息家电和通信设备被广泛使用,嵌入式 Web 服务器则是其中关键的技术设备。本文介绍了 Web 服务器的结构设计,并对系统中采用的关键技术及其实现进行了论述。

**关键词**：嵌入式;Web 服务器

**中图分类号**：TP393 **文献标识码**：A

## Design and research of the Web server for embedded systems

REN Bin

(Jiangsu Food Science College, Huai'an 223003, China)

**Abstract**: With the rapid development of computer technology and network technology, the embedded device-based monitoring systems, information appliances and communication equipment are widely used, embedded Web server is one of the key technology and equipment. This paper introduces the structural design of Web servers and discusses the key technology and its implementation used in the system.

**Key words**: embedded; Web server

Web 应用程序与传统应用程序相比,具有许多特点和优势,随着 Web 应用程序工具与技术的快速发展,Web 应用程序的应用也越来越广泛。同时,由于 Internet 技术的渗透,嵌入式系统正变得越来越智能化并具有越来越多的网络友好特性,嵌入式 Web 服务器的应用也越来越广泛,它可以广泛地用于各种监控系统、信息家电及智能家居系统、通信设备等领域[1]。

## 1 嵌入式 Web 服务器结构

嵌入式 Web 服务器应有如下一些的设计目标:

(1)借助 HTTP 协议统一并通用化设备或终端的访问接口。

(2)实现对 HTTP 1.0 和 HTTP 1.1 的支持,实现HTTP 的部分方法(GET、POST、HEAD 等),支持 Basic、Digest 加密的认证,支持服务器"推"技术等。

(3)集成简单的应用服务器功能,便于应用开发人员方便地构建基于嵌入式 Web 服务器的应用系统。

(4)支持多种协作接口,嵌入式 Web 服务器应该可以以各种接口方式与相应的模块进行集成,这主要是通过动态库来实现。

(5)支持多种传输方式,嵌入式 Web 服务器需设计一个连接层,该层用来抽象各种具体的传输方式,提供统一和标准的连接服务,Web 服务器本身只依赖一个可靠的传输通道,并不限定是基于 TCP 的。

根据以上嵌入式 Web 服务器的设计目标,Web 服务器的核心在于其 HTTP 引擎和分析引擎,前者主要负责 HTTP 协议请求和响应消息处理,后者用于解析网页中的嵌入式标记,以实现动态内容支持。图 1 是基于嵌入 Web 服务器的应用系统框架,图中的浏览器是客户端,用户接口库是嵌入式 Web 服务器和设备的其他控制等部分的接口,由应用开发人员提供。虚线框中的是嵌入式 Web 服务器的框架结构,该设计的基本思想来源于经典的 MVC(模型-视图-控制)模型,这里把 HTTP 引擎和分析引擎作为控制器,待分析网页和静态网页是视图,而用户接口库则是作为模型,专注于业务逻辑。利用控制器来分离模型和视图,实现模块间松散耦合的效果,可以提高系统灵活性、复用性和可维护性[2]。

### 1.1 HTTP 引擎

HTTP 引擎主要负责对客户访问的过滤和权限检

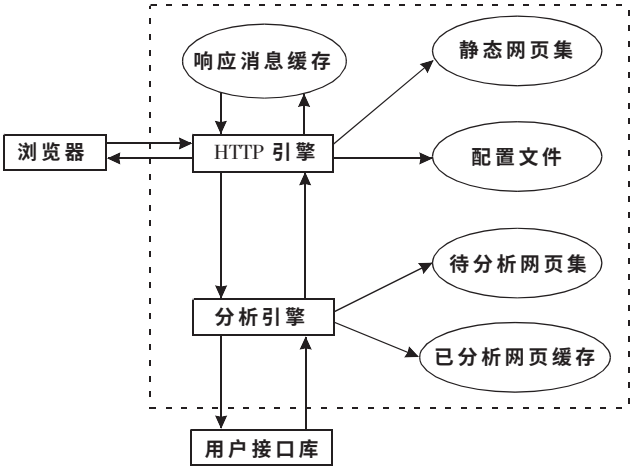


图 1 基于嵌入式 Web 服务器的应用系统框架

查,HTTP 请求和响应消息的接收和发送,会话(session)管理,客户认证和授权,消息的封装、解析和消息安全性检查。

首先,HTTP 引擎进行初始化工作,为该设备站点初始化相应的配置数据结构,如设置站点基本信息、建立用户账号、初始化各目录的访问权限、初始化嵌入式标记的映射定义等。然后 HTTP 引擎接收请求连接,提取访问客户的客户端的信息(如 IP 等),根据配置文件的站点过滤设置进行筛选,如不允许,则返回出错页面,结束处理。

为了加强嵌入式应用的安全管理,嵌入式 Web 服务器默认设置:假定任何访问客户均需提供用户名和口令,接着进行客户认证,在客户会话超时后的请求或初次发送请求时会要求输入客户的用户名和密码,一般在浏览器被关闭前或会话超时前,用户再次访问时不需再次提供,除非他所请求的操作需要更高的权限。客户的用户名和密码是通过 HTTP 的 401 Unauthorized 响应头部来激发客户端的浏览器提示用户输入的。

认证通过后,HTTP 引擎根据配置文件的定义给该客户赋予相应的角色和权限,并设立客户会话环境,便于跟踪客户的访问。

当 1 个 HTTP 请求被接收后,进入 HTTP 引擎的处理过程。根据资源 URI 判断,若客户请求的是静态网页(网页的后缀是 htm 或 html),那么 HTTP 引擎在静态网页集中查找,加载到内存,并按照 HTTP 的协议规范封装响应消息,包括各头部和协议实体(响应的数据部分)。另外,若本网页是采用服务器"推"技术的,其头部信息是不同的,并与客户端保持永久连接,此后不断地刷新客户端的显示内容。HTTP 引擎将封装好的响应消息返回给客户端,并刷新缓存。

### 1.2 分析引擎

分析引擎专注于待分析网页的分析处理,实现动态页面。其实现的传统技术思路有以下两种方法[3]:

(1)采用传统的 CGI 方式。该方式所需的资源开销较大,每当用户请求 CGI 脚本时,嵌入式 Web 服务器必须初始化 CGI 的运行环境,导入有关参数,然后启动其他的进程来运行 CGI 代码,运行结束后,再释放进程的相关资源,服务器需要承担所有额外的负担,而且和动态库的接口也比较麻烦。

(2)使用 PHP 等脚本语言构建类似主机系统的 Web 应用服务器。当一个访问客户打开网页时,服务端便执行 PHP 的命令,并将执行结果发送至访问者的浏览器中。该方案与台式机系统有最好的兼容性,可以充分借鉴和使用台式机系统的现有软件和相关源代码。但是,该方案的实现需要很多资源,需要构建相关的 PHP 库,而 PHP 和嵌入式系统的接口比较困难,因为 PHP 的设计本身是作为一种服务器端的嵌入式 HTML 的脚本语言,让它直接和驱动级的动态库等进行链接,实现较复杂,特别是不能很好地控制它们之间的交互。而嵌入式系统最需要的是一种实现成本小,而且比较简单、易于使用、具有相当的灵活性和可控性,并且和动态库有很好的配合机制。

上述两种方式还有一个共同的缺点,就是实现业务逻辑的代码经常和网页的布局代码交叉在一起,严重影响程序的可读性、可维护性和可扩展性。由于属于不同层次的代码混在一起,会使它们的代码的相关性大大提高,从而极不利于系统的维护。

鉴于上述问题,在嵌入式 Web 服务器中设计基于嵌入式标记的 HTML 网页分析引擎。采用该方法设计的分析引擎的基本特点如下:

(1)视图(页面布局)、控制(分析引擎)、模型(接口库)松耦合。页面布局和接口库可以分别独立地进行设计和开发,前者专注于 HTML 页面的布局和美化设计,后者则侧重于和控制单元的交互,而分析引擎主要是协调两者的工作。通过待分析网页中的嵌入式标记,关联由应用开发人员编写动态库中的某个函数,以此来动态生成网页。

(2)分析引擎中集成了若干应用开发人员常用的基本功能,如用户权限管理和口令认证、支持缓存机制、多级客户授权、一定的安全防范措施等。在此基础上,应用开发人员可以只专注于业务逻辑的处理和调整相关配置,就能构建和灵活调整配置、功能丰富的基于嵌入式 Web 服务器的系统。

分析引擎和 HTTP 引擎是通过共享 HTTP 请求和响应队列来通信的。在分析引擎的每个运行周期中,进行如下工作:查看 HTTP 请求队列是否为空,若有任务则进行分析处理,然后将结果挂入和 HTTP 引擎共享的响应队列。分析引擎是周期性运行的,将其挂入运行队列后,即可处理下个周期的请求。如果循环 1 个周期而没有任务或结果,则进入睡眠。一般分析引擎是单线程,因为在它的运行过程中不大会出现阻塞。此外,作为嵌入式

Web 服务器，同时访问量一般不大，即使采用多线程也不大可能显著地提高系统的性能，而且多线程有更多的开销。但某些应用可能需要多线程来提高系统的性能。

### 1.3 嵌入式标记

嵌入式标记的语法是任意连续的以空格结尾的数字或字符(除空格和圆括号)序列，设计时采用易于书写和记忆的简短的字母数字序列。共有两类嵌入式标记在应用开发人员设计的待分析网页和静态网页时使用。

(1)参数替换标记。该类标记用于将传入的 POST 等请求的参数的值直接替换网页中相应的参数替换标记。这类标记既是 HTTP 请求的参数，又可能是结果替换标记的参数。

(2)结果替换标记。该类标记用于将其代表的代码段执行结果置换网页中的相应结果替换标记，在置换时可能要进行类型转换。例如，返回值是整型(int)4 000，引擎会将其转换成字符串“4 000”，然后替换。

### 1.4 外部接口库

外部接口库是由应用开发人员提供的用于进行业务逻辑处理的动态库中的某些函数构成的，这些函数(称为外部函数)遵循下面描述的规则：配置文件中的映射定义部分允许应用开发人员灵活地将外部函数的运行特性和与待分析网页中的结果替换标记的进程对应。分析引擎通过收集的外部函数的参数信息和实参值，加载指定的动态库，搜索该外部函数的入口地址，然后构造实际的函数调用，再采集其返回值，替换相应标记，形成动态网页。

这种思路的基本实现技术是动态装载。过程如下：分析引擎在第 1 次调用该动态库中的函数时将其加载(dlopen)到运行进程的地址空间，在库中查找(dlsym)函数的地址，然后调用该函数，当不再需要的时候，卸载(dlclose)动态库。

在嵌入式 Web 服务器启动时，准备了另外的运行线程池，专门用于运行应用开发任务提供的函数，该线程受嵌入式 Web 服务器的分析引擎控制。之所以要在另外的线程中运行，是为尽量避免外部函数代码对本引擎的影响，可以在出现异常时果断地终止该运行线程，加上超时定时器，能将外部函数带来的风险降低。

运行线程和分析引擎之间通过共享运行队列和结果队列来通信。一旦分析引擎有执行函数的需求时，就封装 1 个执行结构 run_struct，它定义了某次 HTTP 请求所需执行的所有函数及其参数、运行超时时间等信息，然后将该 run_struct 挂入运行队列。运行线程是定期访问运行队列的，发现有任务时，就加载相应的共享库，查找函数的入口地址，并加以执行，等待其执行结束，填充结果，直到 run_struct 中的所有函数执行完毕(若期间某函数执行异常，有可能导致提前结束)，最后将运行完成的 run_struct 从运行队列上摘下，挂入结果队列等待分析引擎的访问。

运行线程是等待外部函数执行完毕后，再继续执行的。如果有多个 HTTP 请求等待处理，而又只有 1 个运行线程时，可能会有较大的延时，故可根据具体应用的实际情况，将运行线程的数量配置成多个。但不可过多，因为线程本身也要占有一定的资源，而线程切换同样需要开销。

## 2 嵌入式 Web 服务器的设计

嵌入式 Web 服务器与通用 Web 服务器比较，存在以下 3 个方面的差别[4]：

(1)运行的目标环境不一样。通用 Web 服务器一般运行在计算资源和内存资源都比较丰富的台式机上，而嵌入式 Web 服务器运行的目标系统大多是各类专用设备，资源比较缺乏。

(2)在各自系统中的作用不一样。通用 Web 服务器主要是利用 Web 服务器向用户提供信息服务，而嵌入式 Web 服务器嵌入在设备中，其主要作用是控制和配置设备，但也向客户提供设备的运行信息。

(3)运行的优先级不一样。嵌入式 Web 服务器作为一种监控、管理手段存在，它不能干扰设备主要任务的运行。

针对于上面嵌入式 Web 服务器与通用 Web 服务器的不同，设计嵌入式 Web 服务器时考虑的性能指标也不一样。对于嵌入式 Web 服务器来说，吞吐率并不需要很大，而需要很快的系统响应速度，因为嵌入式 Web 服务器面对的用户是少量的设备管理人员，实现的是监控与管理功能。一个好的 Web 服务器应具备良好的可移植性、可裁剪性和与目标设备良好的兼容性。

嵌入式 Web 服务器中，其 Web 引擎 (HTTP 引擎和分析引擎)是核心部分。嵌入式 Web 引擎的状态图如图 2 所示。

图 2 中的管理应用是嵌入式 Web 服务器与应用设备的接口。它用来实现获取用户所需的设备信息及执行实际的管理设备功能，构成监控、管理设备的用户界面。

有时该用户界面是静态的，但大部分是随时间动态改变的。如客户请求设备状态参数时，嵌入式 Web 服务器通过调用管理应用程序获得设备动态信息，然后组织成动态 Web 页面返回给客户端；如用户通过 Web 页面发送 1 个控制命令字，Web 服务器也需通过调用管理应用程序，把该命令相关参数传递给实际执行控制动作的管理程序并监视其执行，把执行结果组织成 Web 页面的形式返回给用户。在嵌入式 Web 服务器中实现应用程序的接口技术有公共网关技术(CGI)和服务器端包含技术(SSI)。

嵌入式系统中，Web 服务器作为单独的任务执行，任务的优先级应设置成较低的优先级，以避免对嵌入式系统主要功能造成干扰。嵌入式 Web 服务器的结构包

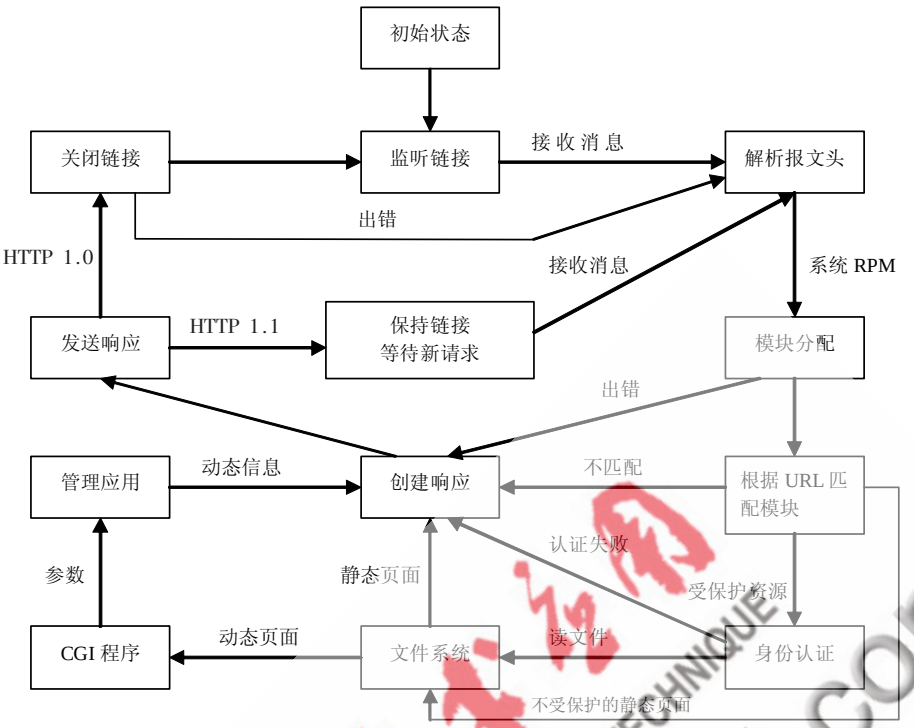


图 2 嵌入式 Web 引擎的状态图

括核心部分和可裁剪的功能模块部分。核心部分包含 HTTP 请求分析引擎和模块分析器。HTTP 请求分析引擎负责接收客户端发送来的 HTTP 请求报文，获得客户端信息，并把解析出的信息保存到请求结构中，后续处理模块都从该结构中获得请求报文的相关信息；模块分析器则是根据配置信息调度其他功能模块。模块分配器执行流程如图 3 所示。

RPM(Request Process Module)可以根据用户的需求进行裁剪，它分成两种类型：系统 PRM 和用户 PRM。一旦配置了系统 PRM 模块，则该 PRM 模块对服务器接收到的所有请求都会进行处理；用户 PRM 模块则根据进一步的配置信息，只对某些特定的请求进行处理。

针对嵌入式 Web 服务器采用以下措施可以提高系统性能：

(1)减小 Web 服务器中请求的生命周期以缩短等待时间。

(2)在 Web 服务器中使用线程取代进程来加强系统的并发性。

(3)采用瘦 Web 服务器、胖客户机的应用模式以提升服务性能。

随着因特网的迅速发展，需构造嵌入式 Web 应用系统的应用也越来越多，嵌入式 Web 服务器将会有很好的发展前景。

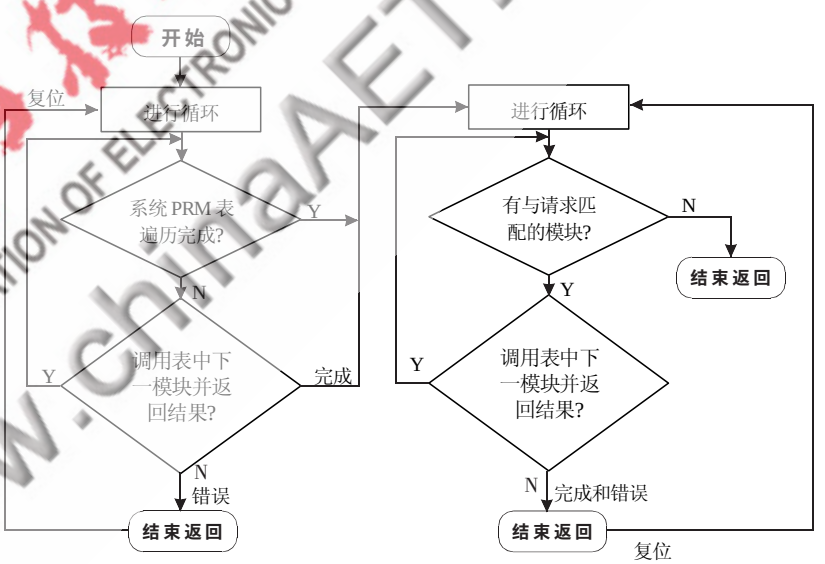


图 3 模块分配器执行流程图

参考文献

[1] 冀振燕.UML 系统分析设计与应用案例[M].北京:人民邮电出版社,2003.

[2] CONALLEN J. Building Web applications with UML. Addison Wesley Longman,2000.

[3] 陈伟,邱书波.蓄电池生产控制网络结构的设计与实现[J].微计算机信息,2009(1-3).

[4] DOUGLASS B P.实时 UML——开发嵌入式系统高效对象[M].尹浩琼译.北京:中国电力出版社,2003.

(收稿日期:2009-05-07)