

(12) 发明专利申请

(10) 申请公布号 CN 102185903 A

(43) 申请公布日 2011.09.14

(21) 申请号 201110097547.4

(22) 申请日 2011.04.19

(71) 申请人 北京神州数码思特奇信息技术股份有限公司

地址 100085 北京市海淀区上地九街9号数码科技广场二层

(72) 发明人 段嘉

(74) 专利代理机构 北京轻创知识产权代理有限公司 11212

代理人 杨立

(51) Int. Cl.

H04L 29/08 (2006.01)

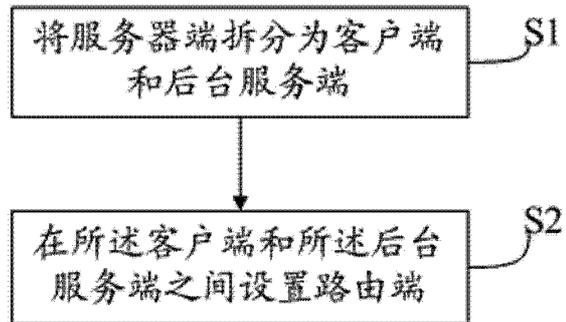
权利要求书 1 页 说明书 4 页 附图 1 页

(54) 发明名称

一种使 B/S 系统具有分布式特性的方法

(57) 摘要

本发明涉及一种使 B/S 系统具有分布式特性的方法,包括步骤 S1、将 B/S 系统中的服务器端拆分为客户端和后台服务端,所述后台服务端包括子服务器;步骤 S2、在所述客户端和所述后台服务端之间设置路由端,该路由端包括路由信息表。本发明使传统 B/S 系统实现了分布式部署,而且实现过程不需修改 B/S 系统的大量代码,只需要将原有系统中的配置文件做简单的修改即可轻松完成,节省了大量的人力物力,有效控制了优化成本。



1. 一种使 B/S 系统具有分布式特性的方法,其特征在于,包括:
步骤 S1、将 B/S 系统中的服务器端拆分为客户端和后台服务端,所述后台服务端包括子服务器;
步骤 S2、在所述客户端和所述后台服务端之间设置路由端,该路由端包括路由信息表。
2. 根据权利要求 1 所述的使 B/S 系统具有分布式特性的方法,其特征在于,步骤 S1 还包括以下子步骤:
S11、将业务处理模块部署到所述子服务器中。
3. 根据权利要求 1 所述的使 B/S 系统具有分布式特性的方法,其特征在于,步骤 S1 还包括以下子步骤:
S12、在所述客户端设置代理机制;
S13、在所述后台服务端设置反射机制。
4. 根据权利要求 2 所述的使 B/S 系统具有分布式特性的方法,其特征在于,步骤 S2 还包括以下子步骤:
S21、将所述业务处理模块在所述子服务器上的部署信息存储在所述路由信息表中。
5. 根据权利要求 3 所述的使 B/S 系统具有分布式特性的方法,其特征在于,所述代理机制为动态代理方式。
6. 根据权利要求 5 所述的使 B/S 系统具有分布式特性的方法,其特征在于,所述动态代理方式采用的是 Cglib。
7. 根据权利要求 2 或 4 所述的使 B/S 系统具有分布式特性的方法,其特征在于,所述步骤 S11 还包括以下子步骤:
S110、将每种所述业务处理模块部署到至少两个所述子服务器中。
8. 根据权利要求 4 所述的使 B/S 系统具有分布式特性的方法,其特征在于,所述部署信息包括业务处理模块的域名称、承载的业务标识、所在子服务器信息。

一种使 B/S 系统具有分布式特性的方法

技术领域

[0001] 本发明涉及 B/S 系统,尤其是一种使 B/S 系统具有分布式特性的方法。

背景技术

[0002] 在目前的电信行业,运营支撑系统业务服务运营多为 B/S (浏览器端 / 服务器端) 结构,一般是利用 Java EE 进行开发的,其中的 Java EE 是 Java 的一种标准,是 Java 平台企业版的简称(Java Platform, Enterprise Edition),用于开发便于组装、健壮、可扩展、安全的服务器端 Java 应用。传统 B/S (浏览器端 / 服务器端) 系统工作过程是用户利用浏览器端访问业务,由服务器端对不同业务进行处理后返回给对应的用户所在浏览器端从而完成信息的传递。

[0003] 可见在传统的 B/S 结构中,业务处理中的只有小部分事务逻辑是在浏览器端实现的,大部分的事务逻辑都是在服务器端实现的,根本谈不上负载均衡。同时,随着电信业中业务量的提高,访问量和数据流量的快速增长,电信运营商要求业务处理系统不仅要达到负载均衡,还要具有容灾、可扩展和易于维护的能力,比如当系统内局部出现故障时,可以对故障部分单独进行维护,而不影响其他业务的正常运行;添加新业务功能时,只需对该业务进行单独部署而避免影响系统其它功能的正常运行。以上这些要求都意味着 B/S 系统应当具有分布式特性。

[0004] 但是对现有技术中利用 Java EE 开发的 B/S 系统,添加新的特性通常需要重新添加代码并进行的硬件升级,成本昂贵,耗时严重。

发明内容

[0005] 本发明的目的在于提供一种使 B/S 系统具有分布式特性的方法,实现利用 Java EE 让 B/S 在不修改任何代码的前提下实现分布式特性。

本发明解决上述技术问题的技术方案如下:一种使 B/S 系统具有分布式特性的方法,包括:

步骤 S1、将 B/S 系统中的服务器端拆分为客户端和后台服务端,所述后台服务端包括子服务器;

步骤 S2、在所述客户端和所述后台服务端之间设置路由端,该路由端包括路由信息表。

[0006] 通过将原有 B/S 系统中的服务器端拆分为客户端和后台服务端,使传统 B/S 系统中服务器端在运行时达到服务提供和服务消费解耦合,从而实现分布式。在客户端和后台服务端之间添加路由端可以实现负载均衡。

[0007] 进一步地,步骤 S1 还包括以下子步骤:

S11、将业务处理模块部署到所述子服务器中。

[0008] 将不同业务进一步拆分为更小的业务处理模块,并在后台服务端进行模块化分布,可以提高系统的并行能力。

[0009] 进一步地,步骤 S1 还包括以下子步骤:

S12、在所述客户端设置代理机制；

S13、在所述后台服务端设置反射机制。

[0010] 代理机制和反射机制属于 Java 中的常用方法,代理机制(Proxy)是为其他对象提供一种代理以控制对这个对象的访问,在本发明中相当于客户端和后台服务端之间的网关或转发中介;反射机制(Reflection)是动态获取的信息以及动态调用对象的方法,在本发明中后台服务端可以依靠反射机制调用本地的业务处理模块来完成业务处理。代理机制和反射机制的运用使得客户端与后台服务端之间实现解耦合,并且增加了客户端和后台服务端之间的透明性。

[0011] 进一步地,步骤 S2 还包括以下子步骤:

S21、将所述业务处理模块在所述子服务器上的部署信息存储在所述路由信息表中。

[0012] 通过路由端的路由信息表可以查找到业务请求所对应的业务处理模块的部署信息,为提供路由服务提供保障。

[0013] 优选地,所述代理机制为动态代理方式。

[0014] 动态代理(Dynamic Proxy)是运行时才生成代理的方式,通过这种方式,被代理的对象、需要控制的接口和控制方式都可以在运行时改变,比无法适应运行时的改变的静态代理更具有灵活性。

[0015] 而且,所述动态代理方式采用的是 Cglib。

[0016] Cglib 采用的是 ASM 字节码工具直接修改字节码而产生的代理,不需依赖于代理对象的接口,因此应用限制较少。

[0017] 进一步地,所述步骤 S11 还包括以下子步骤:

S110、将每种所述业务处理模块部署到至少两个所述子服务器中。

[0018] 所述部署信息包括业务处理模块的域名称、承载的业务标识、所在子服务器信息。

[0019] 当业务处理模块所在的其中一个子服务器发生故障时,不会影响其他子服务器上的该业务处理模块对业务的正常处理。

[0020] 通过将原有 B/S 系统中的服务器端拆分为客户端和后台服务端,实现了系统的分布式部署,在客户端和后台服务端之间添加路由端实现了负载均衡,而且实现过程不需修改 B/S 系统的大量代码,只需要将原有系统中的配置文件做简单的修改即可轻松完成,节省了大量的人力物力,有效控制了优化成本。

附图说明

[0021] 图 1 为本发明的方法流程图;

图 2 为本发明所实现的分布式示意图。

具体实施方式

[0022] 以下结合附图对本发明的原理和特征进行描述,所举实例只用于解释本发明,并非用于限定本发明的范围。

[0023] 根据图 1 所示,本发明提供了一种使 B/S 系统具有分布式特性的方法,包括:

步骤 S1、将 B/S 系统中的服务器端拆分为客户端和后台服务端,所述服务端包括子服务器;步骤 S2、在所述客户端和所述后台服务端之间设置路由端,该路由端包括路由信息

表。

[0024] 传统的 B/S 系统中,服务器端既要接收并分析浏览器发送来的用户请求,又要对分析后的用户请求进行处理,返回处理结果。它同时担任了服务消费者和服务提供者这两种角色。二者在传统 B/S 系统中无法互相独立地完成各自的功能,耦合度很高。因此,本发明将二者拆分为两个功能区,由客户端负责接收来自浏览器发送来的用户业务请求,经过客户端将请求信息封装起来,再通过 http 传递到后台服务端,后台服务端则负责接收该业务请求并按请求进行业务处理,将处理结果通过 http 返回给客户端。其中,客户端可以包括多个客户机,不同种类的业务请求可以由不同的客户机进行接收和分析,而后台服务端也可以划分为多个的子服务器,用来分别处理不同种类的业务。比如,在图 2 中分别将经营分析业务、渠道管理业务、用户管理业务由客户端的经营分析客户机、渠道管理客户机、用户管理客户机分别进行接收和分析,在后台服务端由经营分析子服务器、渠道管理子服务器、用户管理子服务器对业务进行分别处理。

[0025] http 是超文本传输协议,是常用的终端之间请求和应答的标准。

[0026] 因此,进一步地,步骤 S1 还包括子步骤 S11、将业务处理模块部署到所述子服务器中。例如,将用户关系业务处理模块部署到承载用户关系子服务器中;将渠道管理业务处理模块部署到承载渠道管理子服务器中。这样就从功能上实现了服务提供者从一到多的分布式部署,降低了传统 B/S 系统服务器端的耦合度。

[0027] 步骤 S2、在所述客户端和所述后台服务端之间设置路由端,该路由端包括路由信息表。步骤 S2 还包括以下子步骤 S21、将所述业务处理模块在所述子服务器上的部署信息存储在所述路由信息表中。其中,部署信息包括业务处理模块的域名称、承载的业务标识、所在子服务器信息。例如,经营分析业务处理模块的域名称为“经营分析”;为该业务处理模块添加一个 Bean 时,所命名的 Bean ID 就是承载的业务标识(其中 Bean 是 Java 中的一个标准,属于公知技术);所在子服务器信息包括业务处理模块部署的子服务器上的地址和子服务器的名称、标识等。子服务器信息与域名称、承载的业务标识之间在路由信息表内具有映射关系。路由端通过路由信息表向客户端提供了向后台服务端传递数据的指向,帮助客户端找到业务请求所应该到达的目的地。

[0028] 进一步地,步骤 S1 还包括以下子步骤:

S12、在所述客户端设置代理机制;

S13、在所述后台服务端设置反射机制。

[0029] 其中,所述代理机制为动态代理方式。代理机制(Proxy)是为其他对象提供一种代理以控制对这个对象的访问;反射机制(Reflection)是动态获取的信息以及动态调用对象的方法的机制,就是在运行状态中,对于任意一个类,都能够知道这个类的所有属性和方法,对于任意一个对象,都能够调用它的任意一个方法。代理机制在本发明中相当于客户端和后台服务端之间的网关或转发中介,客户端通过代理机制将业务请求先传递给路由端,路由端根据业务请求查找路由信息表,轮询返回部署该业务的子服务器信息,然后客户端根据返回的子服务器信息,将业务请求直接发送到后台服务端相应的子服务器上,该子服务器收到该请求后通过反射机制调用相应的业务处理模块,将处理结果数据传回给客户端并通知用户所在的浏览器端。后台服务端上的子服务器传递业务请求,后台服务端也通过代理机制向客户端返回业务处理结果。

[0030] 代理根据创建时间的不同,分为静态代理和动态代理(Dynamic Proxy),静态代理是由程序员创建或特定工具自动生成源代码,在程序运行前,代理类的.class文件就已经存在了,动态代理(Dynamic Proxy)是程序运行时动态创建,通过这种方式,被代理的对象、需要控制的接口和控制方式都可以在运行时改变,比无法适应运行时的改变的静态代理更具有灵活性,而且无需程序员手工编写它的代码,省去了编程工作,而且提高了可扩展性,符合实现分布式特性的要求。

[0031] 其中,动态代理优选地采用的是Cglib。Cglib采用的是ASM字节码工具直接修改字节码而产生的代理,Cglib可以代理没有实现接口的继承的类,由于Cglib不是基于接口的,效率较高,使用较灵活,因此应用限制较少。

[0032] 代理机制和反射机制属于Java语言中常用的设计方法,属于本领域技术人员的公知技术。

[0033] 进一步地,所述步骤S11还包括以下子步骤,S110、将每种所述业务处理模块部署到至少两个所述子服务器中。通过服务分发的方式来分配业务处理请求,即使其中个别子服务器出现故障,也不会影响该业务在其他子服务器上的正常处理能力。

[0034] 上文中的耦合可以理解为实体与实体之间的依赖性。相互依赖性较强就是紧耦合,反之就是松耦合。

[0035] 以上所述仅为本发明的较佳实施例,并不用以限制本发明,凡在本发明的精神和原则之内,所作的任何修改、等同替换、改进等,均应包含在本发明的保护范围之内。

