



(12) 发明专利申请

(10) 申请公布号 CN 104394204 A

(43) 申请公布日 2015. 03. 04

(21) 申请号 201410645860. 0

(22) 申请日 2014. 11. 12

(71) 申请人 浪潮(北京)电子信息产业有限公司
地址 100085 北京市海淀区上地信息路2号
2-1号C栋1层

(72) 发明人 李有超 王渭巍

(74) 专利代理机构 北京安信方达知识产权代理有限公司 11262
代理人 王丹 李丹

(51) Int. Cl.
H04L 29/08(2006. 01)
H04L 12/803(2013. 01)

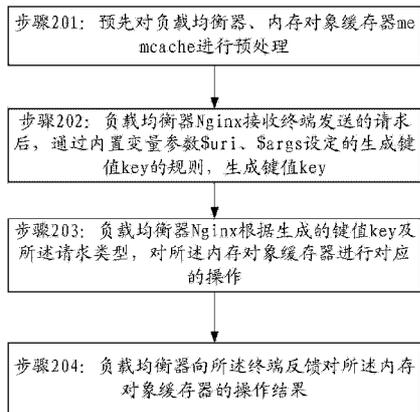
权利要求书2页 说明书3页 附图1页

(54) 发明名称

一种实现负载均衡中信息交互方法及系统

(57) 摘要

本发明提供一种实现负载均衡中信息交互方法及系统。上述方法包括以下步骤:预先对负载均衡器、内存对象缓存器进行预处理;所述负载均衡器接收终端发送的请求后,通过内置变量参数设定的生成键值规则,获取键值;所述负载均衡器根据所述键值及请求类型,对所述内存对象缓存器进行对应的操作。根据本发明提供的一种信息交互方法及系统,实现了终端请求处理的过程中,不需要运行 PHP,这大大提高了请求处理效率,节约了时间资源。



1. 一种实现负载均衡中信息交互方法,其特征在于,包括以下步骤:
预先对负载均衡器、内存对象缓存器进行预处理;
所述负载均衡器接收终端发送的请求后,通过内置变量参数设定的生成键值规则,获取键值;
所述负载均衡器根据所述键值及请求类型,对所述内存对象缓存器进行对应的操作。
2. 根据权利要求1所述的方法,其特征在于:所述负载均衡器向所述终端反馈对所述内存对象缓存器的操作结果。
3. 根据权利要求2所述的方法,其特征在于,所述操作结果包括:提取一个保存在内存对象缓存器上的数据、保存数据到内存对象缓存器上、从内存对象缓存器上删除一个保存的项目、获取当前内存对象缓存器运行的状态、刷新所有内存对象缓存器上保存的项目、打开一个到内存对象缓存器的连接、打开一个到内存对象缓存器的长连接、关闭一个内存对象缓存器的连接、替换一个已经存在内存对象缓存器上的项目。
4. 根据权利要求1所述的方法,其特征在于:预先对负载均衡器进行预处理包括:配置 set 命令参数、配置 add 命令参数、配置 delete 命令参数;配置内置变量参数 \$uri、\$args;配置 location 参数、配置缓存失效时间 \$memc_exptime 参数。
5. 根据权利要求4所述的方法,其特征在于:location 参数中包含了 srcache_fetch、srcache_store;srcache_fetch 表示注册一个输入拦截处理器到 location 参数,这个配置将在 location 参数调用时被执行;而 srcache_store 表示注册一个输出拦截器到 location 参数,当 location 参数执行完成并输出时会被执行。
6. 根据权利要求4所述的方法,其特征在于:通过内置变量参数 \$uri、\$args 设定生成键值 key 规则为:键值 key 由统一资源标识符 URI 和 URI 查询参数组成。
7. 根据权利要求1所述的方法,其特征在于:预先对内存对象缓存器 memcache 进行预处理包括:预先在内存对象缓存器 memcache 中安装保活模块 http-upstream-keepalive-module 并将内存对象缓存器 memcache 中的 upstream 配置为持久连接 keep-alive 和保存键值与数据对应关系 key-data。
8. 根据权利要求1所述的方法,其特征在于,所述请求类型包括:提取一个保存在内存对象缓存器上的数据请求;保存数据到内存对象缓存器上请求;从内存对象缓存器上删除一个保存的项目请求;获取当前内存对象缓存器运行的状态请求;刷新所有内存对象缓存器上保存的项目请求;打开一个到内存对象缓存器的连接请求;打开一个到内存对象缓存器的长连接请求;关闭一个内存对象缓存器的连接请求;替换一个已经存在内存对象缓存器上的项目请求。
9. 一种实现负载均衡中信息交互系统,其特征在于,包括终端、负载均衡器、内存对象缓存器;其中,所述终端通过所述负载均衡器与所述内存对象缓存器相连;
预先对负载均衡器、内存对象缓存器进行预处理;
所述负载均衡器,用于接收终端发送的请求后,通过内置变量参数设定的生成键值规则,获取键值;
所述负载均衡器,用于根据所述键值及请求类型,对所述内存对象缓存器进行对应的操作。
10. 根据权利要求9所述的系统,其特征在于,所述负载均衡器,还用于向所述终端反

馈对所述内存对象缓存器的操作结果。

一种实现负载均衡中信息交互方法及系统

技术领域

[0001] 本发明属于信息控制领域,尤其涉及一种实现负载均衡中信息交互方法及系统。

背景技术

[0002] 互联网的兴起给人们带来巨大便利的同时,用户无论处于什么位置都可以通过浏览器获取网页信息即 web 信息,大量高并发访问也给 web 服务器的性能提出了更高要求;为了提升性能,几乎所有互联网应用都有缓存机制,其中,Memcache(内存对象缓存模块)是使用非常广泛的一个分布式内存对象缓存系统,有效的利用缓存加速用户请求一直是人们关注的热点。

[0003] 但是,传统缓存策略仍造成效率低下,因为传统上是通过 PHP(Hypertext Preprocessor,超文本预处理器)操作 memcache 的,要执行 PHP 代码,nginx(engine x,网页服务器)就必然要和 FastCGI(Fast Common Gateway Interface/FastCGI,快速通用网关接口)通信,同时也要进入 PHP 的生命周期,因此 SAPI(The Microsoft Speech API,语音引擎)、PHP Core 文件和 Zend Engine 的一系列逻辑会被执行。

[0004] 图 1 为现有技术中信息交互架构图,包括终端、负载均衡器 Nginx、PHP 管理器、内存对象缓存器 memcache;其中,所述终端通过所述负载均衡器 Nginx 与所述 PHP 管理器相连;所述负载均衡器 Nginx 通过所述 PHP 管理器与所述内存对象缓存器 memcache 相连。上述架构的工作原理如下:

[0005] 步骤 101:所述终端向所述负载均衡器发送请求;

[0006] 步骤 102:所述负载均衡器向所述 PHP 管理器发送请求;

[0007] 步骤 103:所述 PHP 管理器向所述内存对象缓存器发送请求;其中,所述 PHP 管理器调用 PHP 函数,向所述内存对象缓存器发送请求;

[0008] 步骤 104:所述内存对象缓存器向所述 PHP 管理器发送请求响应;

[0009] 步骤 105:所述 PHP 管理器向所述负载均衡器发送数据请求响应;

[0010] 步骤 106:所述负载均衡器向所述终端反馈所述请求响应。

[0011] 由此可知,终端请求处理的过程中,需要运行 PHP,这大大降低了请求处理效率,造成时间资源极大浪费。

发明内容

[0012] 本发明提供一种实现负载均衡中信息交互方法及系统,以解决上述问题。

[0013] 本发明提供一种实现负载均衡中信息交互方法。上述方法包括以下步骤:

[0014] 预先对负载均衡器、内存对象缓存器进行预处理;

[0015] 所述负载均衡器接收终端发送的请求后,通过内置变量参数设定的生成键值规则,获取键值;

[0016] 所述负载均衡器根据所述键值及请求类型,对所述内存对象缓存器进行对应的操作。

[0017] 本发明还提供一种实现负载均衡中信息交互系统,包括终端、负载均衡器、内存对象缓存器;其中,所述终端通过所述负载均衡器与所述内存对象缓存器相连;

[0018] 预先对负载均衡器、内存对象缓存器进行预处理;

[0019] 所述负载均衡器,用于接收终端发送的请求后,通过内置变量参数设定的生成键值规则,获取键值;

[0020] 所述负载均衡器,用于根据所述键值及请求类型,对所述内存对象缓存器进行对应的操作。

[0021] 相较于先前技术,根据本发明提供一种实现负载均衡中信息交互方法及系统,实现了终端请求处理的过程中,不需要运行 PHP,这大大提高了请求处理效率,节约了时间资源。

附图说明

[0022] 此处所说明的附图用来提供对本发明的进一步理解,构成本申请的一部分,本发明的示意性实施例及其说明用于解释本发明,并不构成对本发明的不当限定。在附图中:

[0023] 图 1 为现有技术中信息交互架构图;

[0024] 图 2 为本发明的实施例 1 的实现负载均衡中信息交互方法流程图;

[0025] 图 3 所示为本发明实施例 2 的实现负载均衡中信息交互系统结构图。

具体实施方式

[0026] 下文中将参考附图并结合实施例来详细说明本发明。需要说明的是,在不冲突的情况下,本申请中的实施例及实施例中的特征可以相互组合。

[0027] 图 2 为本发明的实施例 1 的实现负载均衡中信息交互方法流程图,包括以下步骤:

[0028] 步骤 201:预先对负载均衡器、内存对象缓存器 memcache 进行预处理;

[0029] 1、预先对负载均衡器进行预处理包括:配置 set 命令参数、配置 add 命令参数、配置 delete 命令参数;配置内置变量参数 \$uri、\$args;配置 location 参数、配置缓存失效时间 \$memc_exptime 参数。

[0030] location 参数中包含了 srcache_fetch、srcache_store;srcache_fetch 表示注册一个输入拦截处理器到 location 参数,这个配置将在 location 参数调用时被执行;而 srcache_store 表示注册一个输出拦截器到 location 参数,当 location 参数执行完成并输出时会被执行。

[0031] 通过内置变量参数 \$uri、\$args 设定生成键值 key 规则,例如:键值 key 由统一资源标识符 URI 和 URI 查询参数(放在 URI 中传递的参数)组成。

[0032] 2、预先对内存对象缓存器 memcache 进行预处理包括:

[0033] 预先在内存对象缓存器 memcache 中安装保活模块 http-upstream-keepalive-module 并将内存对象缓存器 memcache 中的 upstream 配置为持久连接 keep-alive;

[0034] 保存键值与数据对应关系 key-data。

[0035] 步骤 202:负载均衡器 Nginx 接收终端发送的请求后,通过内置变量参数 \$uri、

\$args 设定的生成键值 key 的规则,生成键值 key ;

[0036] 请求类型包括:提取一个保存在内存对象缓存器上的数据请求;保存数据到内存对象缓存器上请求;从内存对象缓存器上删除一个保存的项目请求;获取当前内存对象缓存器运行的状态请求;刷新所有内存对象缓存器上保存的项目请求;打开一个到内存对象缓存器的连接请求;打开一个到内存对象缓存器的长连接请求;关闭一个内存对象缓存器的连接请求;替换一个已经存在内存对象缓存器上的项目请求。

[0037] 步骤 203:负载均衡器 Nginx 根据生成的键值 key 及所述请求类型,对所述内存对象缓存器进行对应的操作;其中,所述内存对象缓存器保存了键值与数据对应关系 key-data。

[0038] 步骤 204:负载均衡器向所述终端反馈对所述内存对象缓存器的操作结果。

[0039] 操作结果包括:提取一个保存在内存对象缓存器上的数据;保存数据到内存对象缓存器上;从内存对象缓存器上删除一个保存的项目;获取当前内存对象缓存器运行的状态;刷新所有内存对象缓存器上保存的项目;打开一个到内存对象缓存器的连接;打开一个到内存对象缓存器的长连接;关闭一个内存对象缓存器的连接;替换一个已经存在内存对象缓存器上的项目。

[0040] 图 3 所示为本发明实施例 2 的实现负载均衡中信息交互系统结构图,包括终端、负载均衡器、内存对象缓存器;其中,所述终端通过所述负载均衡器与所述内存对象缓存器相连;

[0041] 预先对负载均衡器、内存对象缓存器进行预处理;

[0042] 所述负载均衡器,用于接收终端发送的请求后,通过内置变量参数设定的生成键值规则,获取键值;

[0043] 所述负载均衡器,用于根据所述键值及请求类型,对所述内存对象缓存器进行对应的操作。

[0044] 所述负载均衡器,还用于向所述终端反馈对所述内存对象缓存器的操作结果。

[0045] 相较于先前技术,根据本发明提供的一种实现负载均衡中信息交互方法及系统,实现了终端请求处理的过程中,不需要运行 PHP,这大大提高了请求处理效率,节约了时间资源。

[0046] 以上所述仅为本发明的优选实施例而已,并不用于限制本发明,对于本领域的技术人员来说,本发明可以有各种更改和变化。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。

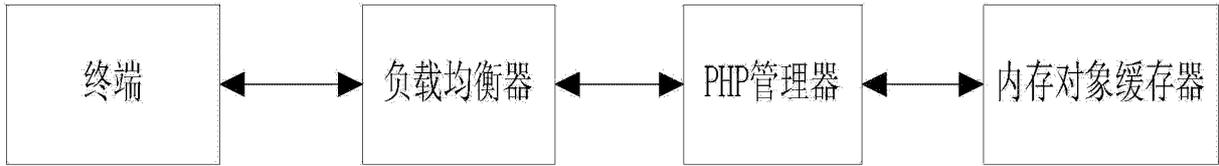


图 1

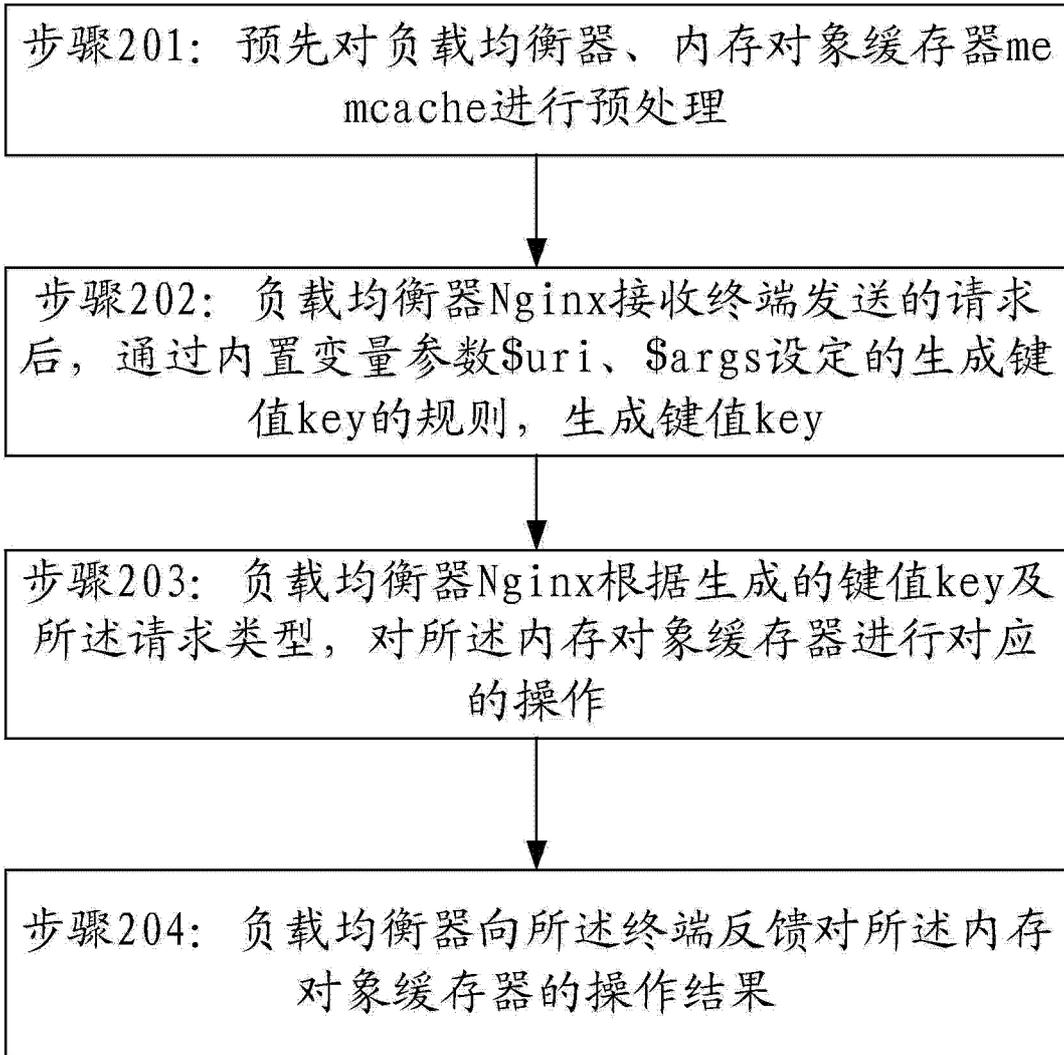


图 2

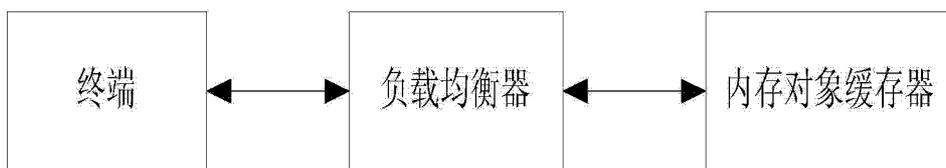


图 3