

[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/46 (2006.01)



[12] 发明专利申请公布说明书

[21] 申请号 200710086918.2

[43] 公开日 2008 年 9 月 24 日

[11] 公开号 CN 101271402A

[22] 申请日 2007.3.20

[21] 申请号 200710086918.2

[71] 申请人 华为技术有限公司

地址 518129 广东省深圳市龙岗区坂田华为
总部办公楼

[72] 发明人 张 宇 李建斌 徐志贤

[74] 专利代理机构 北京集佳知识产权代理有限公司

代理人 马 敬 逯长明

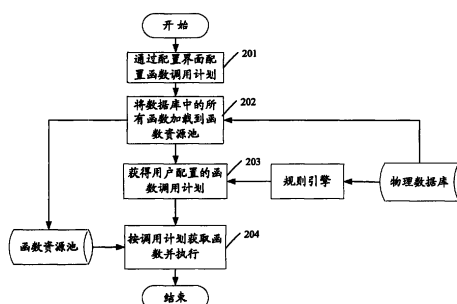
权利要求书 2 页 说明书 6 页 附图 4 页

[54] 发明名称

一种调用函数的方法及系统

[57] 摘要

本发明公开了一种调用函数的方法，包括：配置函数调用计划，将函数动态库中所有的函数加载到函数资源池；加载主控单元，获得所述函数调用计划，根据所述调用计划从所述函数资源池中调用函数并执行。应用本发明，由于调用计划是由用户配置的，因而可以根据用户的需求来控制是否调用动态库函数，以及如何调用等等，再有，由于启动时将函数动态库中的所有函数都加载到函数资源池，避免了运行期内调用函数时动态开启函数动态库以获取函数句柄的性能开销，因而提高了运行期的性能。本发明同时还公开了一种调用函数的系统。



1、一种调用函数的方法，其特征在于，配置函数调用计划并保存，该方法还包括以下步骤：

将函数动态库中所有的函数加载到函数资源池；

获得所述函数调用计划，根据所述调用计划从所述函数资源池中调用函数并执行。

2、根据权利要求1所述的方法，其特征在于，

所述配置函数调用计划的步骤包括：配置用于指示如何调用函数动态库的规则表达式；将所述规则表达式作为配置的函数调用计划；对所述表达式进行编译，生成规则引擎能够识别的编译串，并保存在数据库中；

所述获得函数调用计划的过程包括：从所述数据库中获取规则引擎编译串并执行，根据所述编译串获得函数调用计划。

3、根据权利要求1所述的方法，其特征在于，进一步包括：

根据需求开发新功能函数，将所述新功能函数的基本信息注册到所述函数动态库中；在所述函数调用计划中配置所述新功能函数。

4、根据权利要求3所述的方法，其特征在于，所述新功能函数的开发通过软件开发工具包完成。

5、根据权利要求3所述的方法，其特征在于，所述新功能函数的基本信息包括：函数名、所属动态库名、所述动态库路径、函数的参数个数及参数类型。

6、根据权利要求1所述的方法，其特征在于，所述函数资源池中的被加载函数以函数句柄的形式存在。

7、一种调用函数的系统，其特征在于，包括：

函数动态库，用于存储所有待调用的函数；

配置单元，用于配置函数调用计划并保存；

第一加载单元，用于将函数动态库中的所有函数加载到函数资源池；

函数资源池，用于保存被加载的函数；

主控单元，用于调用所述函数调用计划，根据所述调用计划从所述函数资源池中调用函数并执行。

8、根据权利要求7所述的系统，其特征在于，该系统还包括：

开发单元：用于配置新功能函数，将所述新功能函数的基本信息注册到所述函数动态库中，所述新功能函数根据用户需求开发获得。

9、根据权利要求8所述的系统，其特征在于，所述新功能函数的基本信息包括：函数名、所属动态库名、所述动态库路径、函数的参数个数及参数类型。

10、根据权利要求7所述的系统，其特征在于，所述函数资源池保存的被加载函数以函数句柄的形式存在。

一种调用函数的方法及系统

技术领域

本发明涉及软件开发技术领域，特别涉及一种调用函数的方法及系统。

背景技术

目前大的项目开发中，通常做法是按功能模块分工开发，最后提供功能函数库给上层应用程序调用，形成一个完整的系统。

包含了很多函数和变量的目标代码的文件被称为“库”，库目前分为静态库和动态库。

静态库：其结构比较简单，是将所有目标代码放在一起，链接时链接程序根据每一份目标代码的符号表查找相应的符号(函数和变量的名字)，若查找到则将该函数里面需要定位的符号进行定位，然后将整块函数代码放进可执行文件里，若找不到则报错退出。

使用静态库时，由于链接后产生的可执行文件包含了所有需要调用函数的代码，因此占用磁盘空间比较大；而且，如果有多个调用相同库函数的进程在内存中同时运行，则内存中就存在多份相同的库函数代码，因此占用内存空间比较多；静态库一经改动所有调用此静态库的函数模块都需重新编译。

动态库：其引入就是为了解决静态库的问题所产生的，其加载方式有下面两种：

a) 静态绑定

使用静态绑定的程序在一开始载入内存时，主控程序就会把程序所有调用到的动态代码的地址算出、确定下来，也即启动时将所有可能用到的函数加载到内存。这种方式使得程序的初始化比较长，不过初始化成功后，程序的运行速度比较快。

b) 动态绑定

使用这种方式的程序并不在一开始就完成动态链接，而是直到真正调用动态库代码时，主控程序才计算动态代码的逻辑地址。

目前最常用的是使用动态库的动态绑定方式，因为其可以实现调用的灵活性。即在调用处用 `dlopen()` 函数开启动态库，并用 `dlsym()` 函数调出动态库中的函数符号进行使用，使用完毕后用 `dlclose()` 函数关闭动态库。下面以 Unix 环境下，需调用 `test()` 函数为例进行说明，参见图 1，其是现有应用动态绑定的方式调用函数的流程示意图。

步骤 101，加载主控程序。

步骤 102，通过 `dlopen()` 函数获取调用 `libtest.so` 动态库的句柄 A。

步骤 103，根据句柄 A 和待调用的 `test()` 函数名，用 `dlsym()` 函数获取 `test()` 函数对应的句柄 B。

步骤 104，根据句柄 B 执行 `test()` 函数，执行完毕后调用 `dlclose()` 函数关闭此句柄。

至此，实现动态绑定方式下的 `test()` 函数的调用。

动态库的动态绑定技术虽然实现了动态库函数的灵活调用，但是在实现过程中，发明人发现至少存在如下问题：

1) 对于用户来说函数的调用并不可控制，因为函数的调用都已经固化在主控程序的代码中不能更改。

2) 由于在调用动态库函数时，需要调用 `dlopen()`，`dlsym()` 等函数获得动态库中的函数句柄，这在运行过程中都比较消耗性能，因此动态绑定使其运行期性能比较低。

3) 无法支持用户的二次开发，这是因为，从上例可以看出 `test()` 函数所在动态库，以及函数的参数信息必须都是已知的，且已经固化在主控程序的代码中才能被调用，如果用户想在主控程序中调用其自己开发的 `newTest()` 函数，就必须修改主控程序，并重新编译其主控程序的执行程序，并发布新版本，这对于开发商来说是不现实的。

发明内容

本发明实施例提供了一种调用函数的方法及系统，使得可以根据用户的需求来控制是否调用动态库函数，并提高运行期的性能。

本发明实施例的技术方案包括：

一种调用函数的方法，配置函数调用计划并保存，该方法还包括以下步骤：

将函数动态库中所有的函数加载到函数资源池；

获得所述函数调用计划，根据所述调用计划从所述函数资源池中调用函数并执行。

一种调用函数的系统，包括：

函数动态库，用于存储所有待调用的函数；

配置单元，用于配置函数调用计划并保存；

第一加载单元，用于将函数动态库中的所有函数加载到函数资源池；

函数资源池，用于保存被加载的函数；

主控单元，用于调用所述函数调用计划，根据所述调用计划从所述函数资源池中调用函数并执行。

应用本发明实施例，由于调用计划是由用户配置的，因而可以根据用户的需求来控制是否调用动态库函数，以及如何调用等等，再有，由于启动时将函数动态库中的所有函数都加载到函数资源池，避免了运行期内调用函数时动态开启函数动态库的性能开销，因而提高了运行期的性能。

附图说明

图1是现有应用动态绑定的方式调用函数的流程示意图；

图2是根据本发明实施例的调用函数的流程示意图；

图3是根据本发明实施例的配置函数调用计划的流程示意图；

图4是根据本发明实施例的实现二次开放的流程示意图；

图5是根据本发明实施例的调用函数的系统结构示意图。

具体实施方式

下面结合附图及具体实施例对本发明再做详细说明。

图2所示为根据本发明实施例的调用函数的流程示意图，具体包括：

步骤201，通过配置界面配置函数调用计划并保存。该步骤中的配置由用户来执行，这样就可以根据用户的需求设置调用条件来控制是否调用动态库函数，以及调用哪些函数等等。

步骤202,启动时将函数动态库中的所有函数加载到函数资源池。所述函数资源池中被加载的函数可以以函数句柄的形式存在,也可以以其他形式存在,此处并不对存在的形式加以限制。

步骤203,加载主控单元即主控程序,由主控单元获得用户配置的所述函数调用计划。

步骤204,根据函数调用计划从函数资源池中调用相应函数并执行。

上述配置函数调用计划的具体过程可参见图3,

步骤301,配置用于指示何时需要调用函数动态库的规则表达式;将该规则表达式作为配置的函数调用计划;

步骤302,通过规则引擎对上述表达式进行编译;

步骤303,生成规则引擎能够识别的编译串,将该编译串保存在数据库中。

可以理解,该保存编译串的数据库和上述保存所有函数的函数动态库,在物理上可以是一个实体,在逻辑上可以是两个数据库。

相应的,上述步骤203中的获得函数调用计划的过程包括:从所述数据库中获取规则引擎能够编译串并执行,根据所述编译串获得函数调用计划。

这样,由于调用计划是由用户配置的,因而可以根据用户的需求来控制是否调用动态库函数,以及如何调用等等,再有,由于启动时将函数动态库中的所有函数都加载到函数资源池,避免了运行期内调用函数时动态开启函数动态库以获取函数句柄的性能开销,因而提高了运行期的性能。

此外,本发明实施例还提供了二次开发的功能,即在系统发布后,如果用户提出新的功能需求,可以通过软件开发工具包(SDK, Software Development Kit)包进行功能函数的二次开发,而不需发布新的系统或给系统提供补丁,因而使得系统具有很好的扩展性,对原系统影响非常小。参见图4,其是根据本发明实施例的实现二次开发的流程示意图。具体为:

步骤401,根据需求开发新功能函数,实际应用时,可以利用开放商提供的SDK包进行开发。

步骤402,将所述新功能函数的基本信息注册到所述函数动态库中。所述新功能函数的基本信息包括:函数名、所属动态库名、所述动态库路径、函数的参数个数及参数类型。

这样，在系统启动时，该新的功能函数就可以被加载到函数资源池中，以备后续应用。

步骤403，在所述函数调用计划中配置所述新功能函数的调用计划。由于函数调用计划是由用户配置的，因而，可根据用户的需求加上需调用某个函数，如何调用等等。具体配置过程可参见图3，不再重复说明。

步骤404，加载主控单元即主控程序，获得所述函数调用计划。

步骤405，根据调用计划从函数资源池中调用相应函数并执行。

至此，实现了对新开发函数的调用。而且，该调用是不需修改主控单元即主控程序的。

本领域普通技术人员可以理解，实现上述实施例方法中的全部或部分步骤是可以通过程序来指令相关的硬件来完成，所述程序可以存储于一计算机可读存储介质中，所述存储介质如ROM/RAM、磁盘、光盘等。

本发明实施例还公开了一种调用函数的系统，包括：

函数动态库510，用于存储所有待调用的函数；

配置单元550，用于配置函数调用计划并保存；该具体配置过程可参见前面的图3；

第一加载单元520，用于在启动时将函数动态库中的所有函数加载到函数资源池；

函数资源池530，用于保存被加载的函数；所述函数资源池保存的被加载函数可以以函数句柄的形式存在，也可以以其他形式存在，此处并不对存在的形式加以限制；

主控单元540，用于调用所述函数调用计划，根据所述调用计划从所述函数资源池中调用函数并执行。

该系统还可以包括：开发单元560，用于配置新功能函数，将所述新功能函数的基本信息注册到所述函数动态库中，所述新功能函数根据用户需求开发获得。所述新功能函数的基本信息包括：函数名、所属动态库名、所述动态库路径、函数的参数个数及参数类型。

上述系统还可以包括第二加载单元（图未示），用于在系统启动后首先加载主控单元即主控程序，之后主控单元再执行后续处理。

以上所述仅为本发明的较佳实施例而已，并非用于限定本发明的保护范围。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，均包含在本发明的保护范围内。

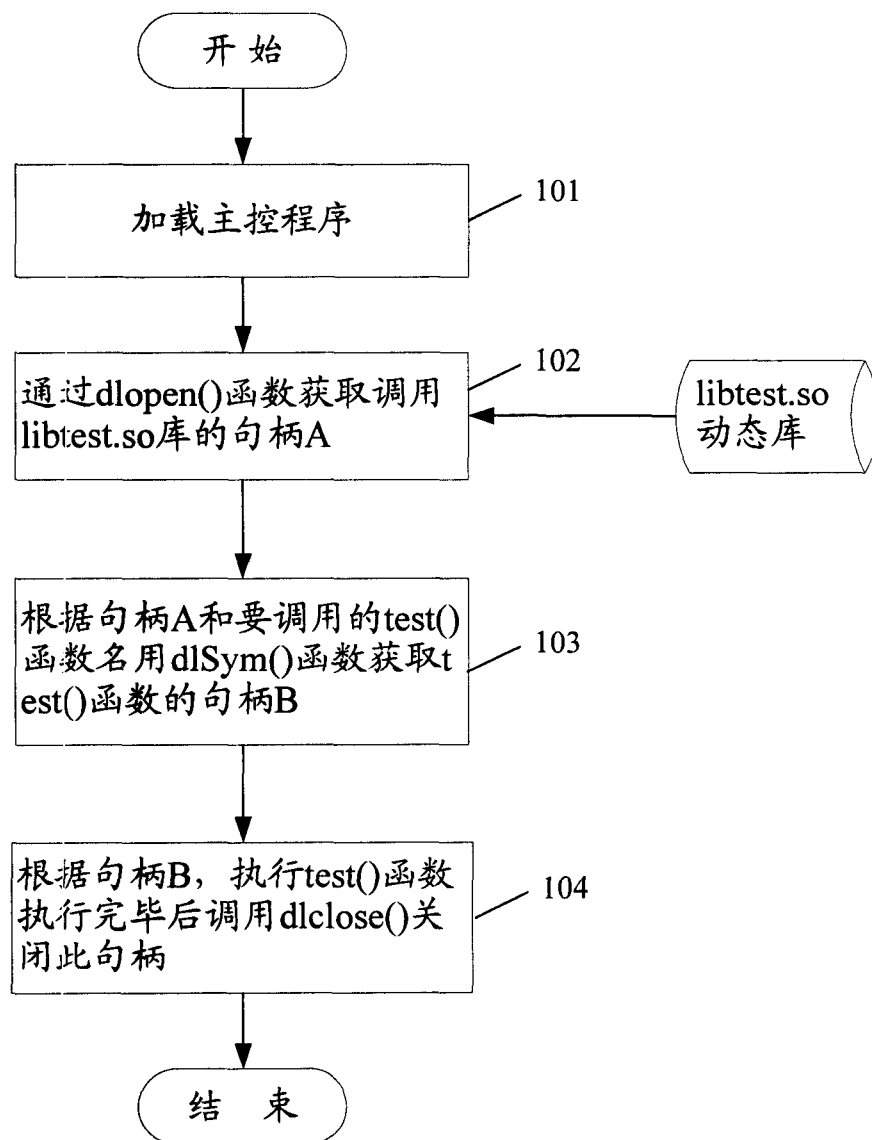


图 1

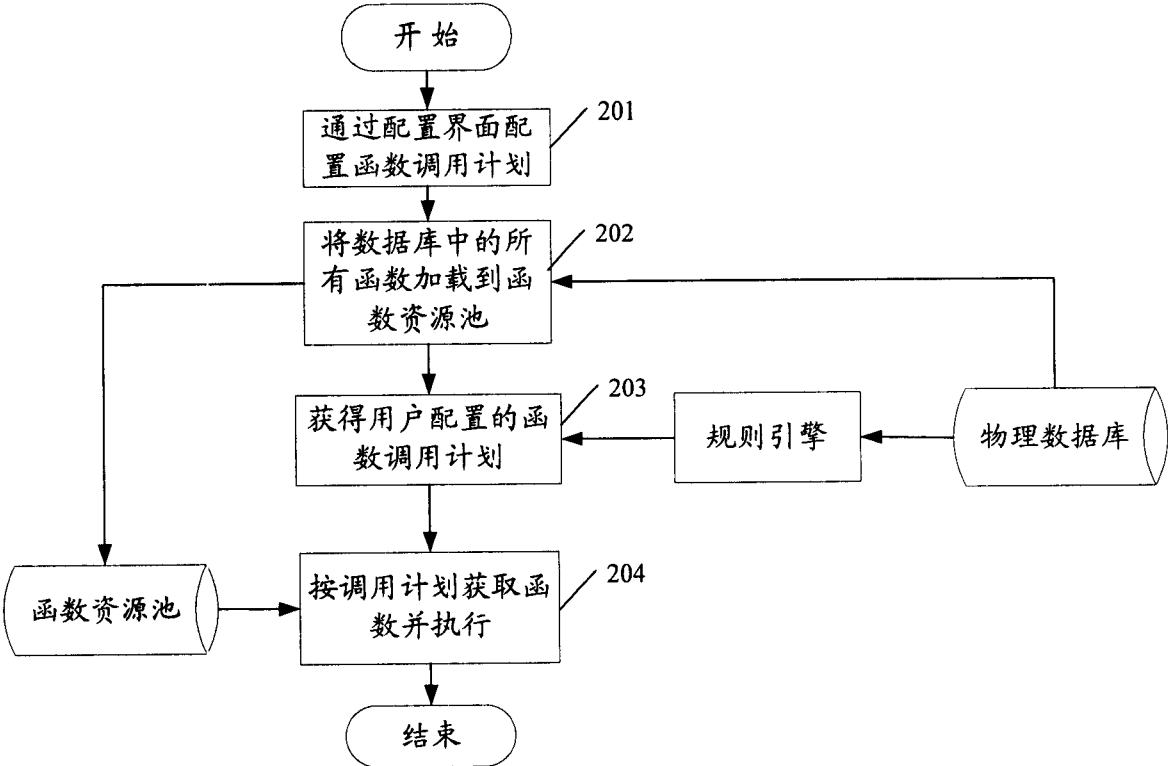


图 2

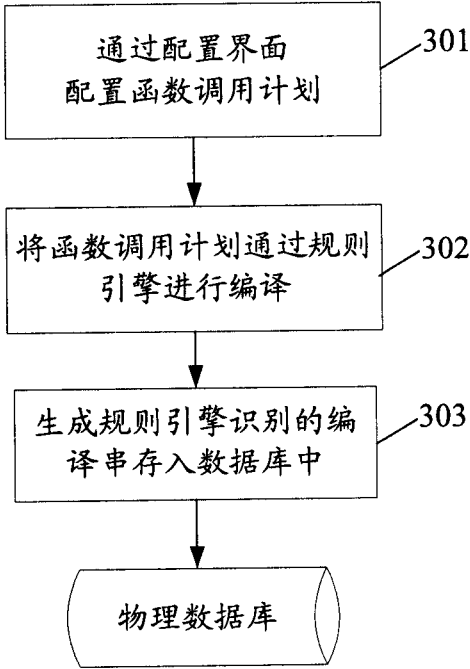


图 3

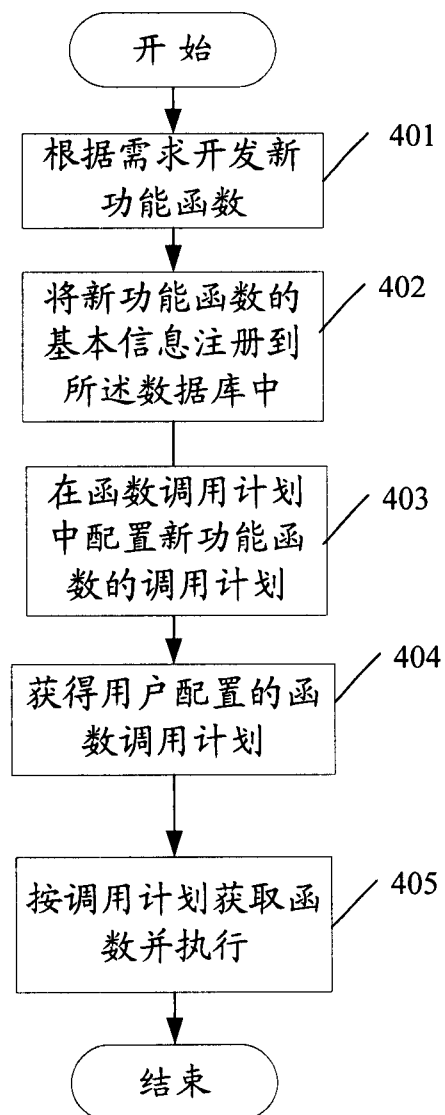


图4

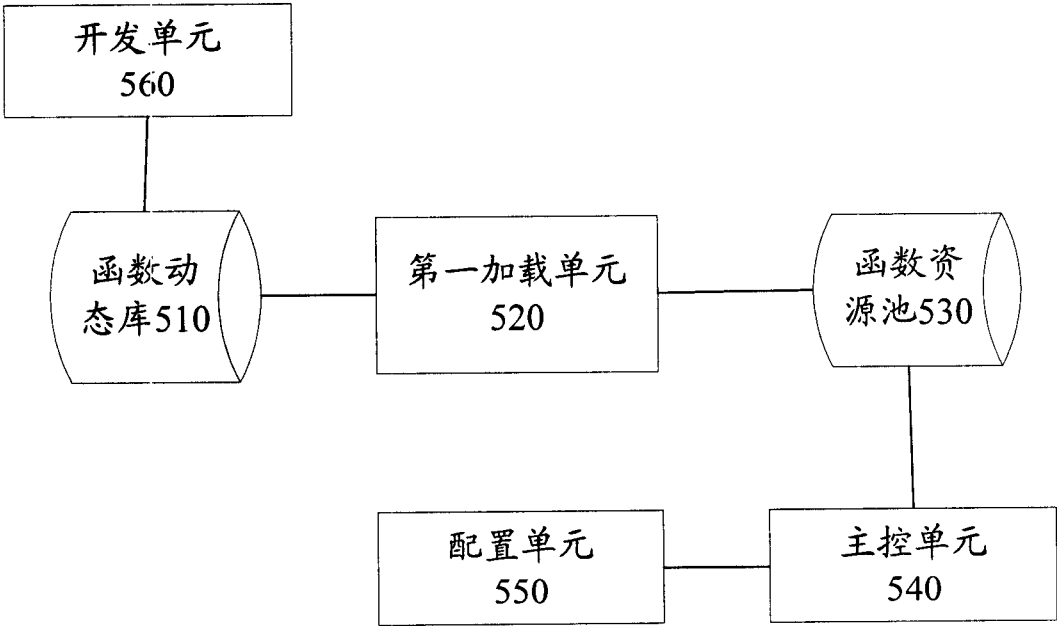


图5