(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 115374017 A
(43)申请公布日 2022.11.22

(21)申请号 202211314934.3

(22)申请日 2022.10.26

(71)申请人 统信软件技术有限公司
地址 100176 北京市大兴区北京经济技术开发区科谷一街10号院12号楼18层

(72)发明人 叶业顺

(74)专利代理机构 北京瀚方律师事务所 11774
专利代理师 姜莹

(51)Int.Cl.
G06F 11/36(2006.01)

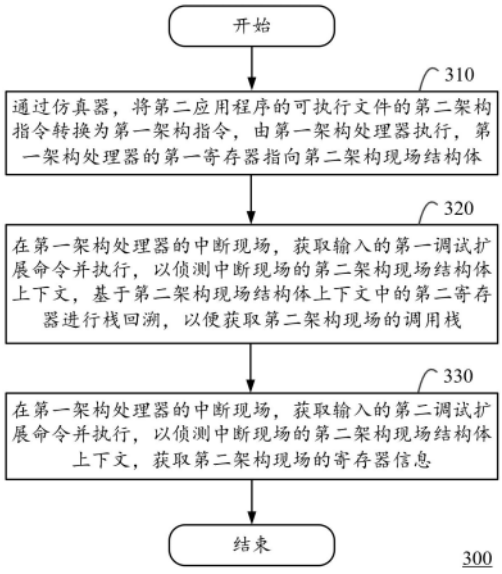
权利要求书2页 说明书9页 附图5页

(54)发明名称

一种仿真运行可执行文件时抓取现场的方法及计算设备

(57)摘要

本发明公开了一种仿真运行可执行文件时抓取现场的方法及计算设备，涉及计算机技术领域。方法包括：通过仿真器将第二应用程序的可执行文件的第二架构指令转换为第一架构指令，由第一架构处理器执行；在第一架构处理器的中断现场，获取输入的第一调试扩展命令并执行，以侦测中断现场的第二架构现场结构体上下文，基于其中的第二寄存器进行栈回溯，以便获取第二架构现场的调用栈；在第一架构处理器的中断现场，获取输入的第二调试扩展命令并执行，以侦测中断现场的第二架构现场结构体上下文，获取第二架构现场的寄存器信息。根据本发明的技术方案，可以实现在第一架构中断现场直观地查看第二架构现场的调用栈和寄存器信息。



CN 115374017 A

1.一种仿真运行可执行文件时抓取现场的方法,在计算设备中执行,所述计算设备中包括第一架构处理器,并运行有第一操作系统;

所述第一操作系统上布置有仿真器以及第二操作系统的兼容层,以便通过所述仿真器仿真运行所述兼容层、并基于兼容层运行第二操作系统的第二应用程序;

所述方法包括:

通过所述仿真器,将所述第二应用程序的可执行文件的第二架构指令转换为第一架构指令,由所述第一架构处理器执行,所述第一架构处理器的第一寄存器指向第二架构现场结构体;

在所述第一架构处理器的中断现场,获取输入的第一调试扩展命令并执行,以侦测所述中断现场的第二架构现场结构体上下文,基于所述第二架构现场结构体上下文中的第二寄存器进行栈回溯,以便获取第二架构现场的调用栈;以及

在所述第一架构处理器的中断现场,获取输入的第二调试扩展命令并执行,以侦测所述中断现场的第二架构现场结构体上下文,获取第二架构现场的寄存器信息。

2.如权利要求1所述的方法,其中,在基于所述第二架构现场结构体上下文中的第二寄存器进行栈回溯之前,包括:

基于所述第一架构处理器的第一寄存器,获取所述第二架构现场结构体。

3.如权利要求1所述的方法,其中,所述兼容层适于加载所述可执行文件,并收集所述兼容层中预先置入的符号信息;在通过所述仿真器,将所述第二应用程序的可执行文件的第二架构指令转换为第一架构指令之前,包括:

所述仿真器接收所述兼容层发送的符号信息;

在获取第二架构现场的调用栈之后,包括:

将所述符号信息与所述调用栈进行匹配,以得到第二架构现场的带符号调用栈。

4.如权利要求3所述的方法,其中,通过所述仿真器,将所述第二应用程序的可执行文件的第二架构指令转换为第一架构指令,包括:

通过调试工具对所述仿真器进行调试,并通过所述仿真器,将所述第二应用程序的可执行文件的第二架构指令转换为第一架构指令。

5.如权利要求4所述的方法,其中,在通过所述仿真器,将所述第二应用程序的可执行文件的第二架构指令转换为第一架构指令之前,包括:

所述仿真器接收所述调试工具发送的获取第二架构现场的请求;

所述方法还包括:

将所述第二架构现场的带符号调用栈和所述第二架构现场的寄存器信息,返回至所述调试工具。

6.如权利要求1-5中任一项所述的方法,其中,通过所述仿真器,将所述第二应用程序的可执行文件的第二架构指令转换为第一架构指令,由所述第一架构处理器执行,还包括:

所述仿真器拦截执行所述第一架构指令时调用的函数,并将所述函数封装后转发到第一操作系统。

7.如权利要求1-5中任一项所述的方法,其中,所述第一寄存器为r0寄存器,所述第二寄存器为ebp寄存器。

8.如权利要求1-5中任一项所述的方法,其中,

所述第一架构为ARM架构,所述第二架构为X86架构;

所述第一操作系统为Linux操作系统,所述第二操作系统为Windows操作系统。

9.一种计算设备,包括:

至少一个处理器;以及

存储器,存储有程序指令,其中,所述程序指令被配置为适于由所述至少一个处理器执行,所述程序指令包括用于执行如权利要求1-8中任一项所述的方法的指令。

10.一种存储有程序指令的可读存储介质,当所述程序指令被计算设备读取并执行时,使得所述计算设备执行如权利要求1-8中任一项所述方法。

一种仿真运行可执行文件时抓取现场的方法及计算设备

技术领域

[0001] 本发明涉及计算机技术领域,尤其涉及一种仿真运行可执行文件时抓取现场的方法及计算设备。

背景技术

[0002] 随着计算机的不断更新换代,操作系统也不断升级。Windows操作系统曾占有了计算机操作系统的绝大部分市场,Windows操作系统上的软件生态也极其丰富。近些年,随着Linux操作系统的兴起,越来越多的计算机选择使用Linux操作系统。因硬件和系统接口不支持等原因,Windows操作系统的软件无法在Linux操作系统上运行。

[0003] 根据CPU架构的不同,在X86、X86_64架构的Linux系统上,可以使用Wine(Wine is not an emulator,通过在X86架构的 Linux系统上实现Windows的系统函数,使得Windows程序在X86架构的Linux系统上可以正常运行的兼容层)来兼容运行Windows操作系统的软件,Windows软件的执行入口是PE文件。但是Wine只提供了X86和X86_64架构的支持,ARM架构无法直接运行Wine。在ARM架构的Linux系统上,需要使用box86来仿真运行Wine。

[0004] 在执行PE文件时,是依靠X86架构的CPU来一条一条地执行对应的X86指令,box86的原理是对X86指令进行解析,翻译成ARM指令来执行,或者,使用c代码模拟X86指令的功能来实现其行为。但是box86仍然存在一些指令未翻译或指令翻译不完善的问题,健壮性不足,这导致在通过box86仿真运行Wine去兼容执行Windows软件的时候,经常会出现运行错误。错误表现可能是:软件执行结果不符合预期、软件执行过程中崩溃、软件执行过程中卡死等。

[0005] 为了解决这些错误,需要使用gdb对box86进行调试。而在对box86的调试过程中,由于X86指令被翻译为ARM指令存放在堆中执行,或模拟成c代码执行,导致出错的现场无法直观的对应到X86指令的错误现场。

[0006] 为此,需要一种仿真运行可执行文件时抓取现场的方法,解决上述方案中存在的问题。

发明内容

[0007] 随着计算机的不断更新换代,操作系统也不断升级。Windows操作系统曾占有了计算机操作系统的绝大部分市场,Windows操作系统上的软件生态也极其丰富。近些年,随着Linux操作系统的兴起,越来越多的计算机选择使用Linux操作系统。因硬件和系统接口不支持等原因,Windows操作系统的软件无法在Linux操作系统上运行。

[0008] 根据CPU架构的不同,在X86、X86_64架构的Linux系统上,可以使用Wine(Wine is not an emulator,通过在X86架构的 Linux系统上实现Windows的系统函数,使得Windows程序在X86架构的Linux系统上可以正常运行的兼容层)来兼容运行Windows操作系统的软件,Windows软件的执行入口是PE文件。但是Wine只提供了X86和X86_64架构的支持,ARM架构无法直接运行Wine。在ARM架构的Linux系统上,需要使用box86来仿真运行Wine。

[0009] 在执行PE文件时，是依靠X86架构的CPU来一条一条地执行对应的X86指令，box86的原理是对X86指令进行解析，翻译成ARM指令来执行，或者，使用c代码模拟X86指令的功能来实现其行为。但是box86仍然存在一些指令未翻译或指令翻译不完善的问题，健壮性不足，这导致在通过box86仿真运行Wine去兼容执行Windows软件的时候，经常会出现运行错误。错误表现可能是：软件执行结果不符合预期、软件执行过程中崩溃、软件执行过程中卡死等。

[0010] 为了解决这些错误，需要使用gdb对box86进行调试。而在对box86的调试过程中，由于X86指令被翻译为ARM指令存放在堆中执行，或模拟成c代码执行，导致出错的现场无法直观的对应到X86指令的错误现场。

[0011] 为此，需要一种仿真运行可执行文件时抓取现场的方法，解决上述方案中存在的问题。

附图说明

[0012] 为了实现上述以及相关目的，本文结合下面的描述和附图来描述某些说明性方面，这些方面指示了可以实践本文所公开的原理的各种方式，并且所有方面及其等效方面旨在落入所要求保护的主题的范围内。通过结合附图阅读下面的详细描述，本公开的上述以及其它目的、特征和优势将变得更加明显。遍及本公开，相同的附图标记通常指代相同的部件或元素。

[0013] 图1示出了根据本发明一个实施例的计算设备100的示意图；

图2示出了根据本发明一个实施例的计算设备100中运行有第一操作系统220的示意图；

图3示出了根据本发明一个实施例的仿真运行可执行文件时抓取现场的方法300的流程示意图；

图4示出了根据本发明一个实施例的仿真运行可执行文件时抓取现场的方法的时序图；

图5示出了根据本发明一个实施例的调用栈的分布示意图。

具体实施方式

[0014] 下面将参照附图更详细地描述本公开的示例性实施例。虽然附图中显示了本公开的示例性实施例，然而应当理解，可以以各种形式实现本公开而不应被这里阐述的实施例所限制。相反，提供这些实施例是为了能够更透彻地理解本公开，并且能够将本公开的范围完整的传达给本领域的技术人员。

[0015] 图1示出了根据本发明一个实施例的计算设备100的示意图。如图1所示，在基本配置中，计算设备100包括至少一个处理单元102和系统存储器160。根据一个方面，取决于计算设备的配置和类型，处理单元102可以实现为处理器。在本发明的实施例中，处理单元102可以实现为第一架构处理器。在一个实施例中，第一架构处理器例如可以为ARM处理器。系统存储器160包括但不限于易失性存储(例如，随机存取存储器)、非易失性存储(例如，只读存储器)、闪速存储器、或者这样的存储器的任何组合。根据一个方面，系统存储器160中包括操作系统150。

[0016] 根据本发明的一个实施例，操作系统150可以实现为第一操作系统，适于运行第一操作系统的应用程序。

[0017] 根据一个方面，操作系统150例如适合于控制计算设备100的操作。此外，示例结合图形库、其他操作系统、或任何其他应用程序而被实践，并且不限于任何特定的应用或系统。在图1中通过在虚线内的那些组件示出了该基本配置。根据一个方面，计算设备100具有额外的特征或功能。例如，根据一个方面，计算设备100包括额外的数据存储设备(可移动的和/或不可移动的)，例如磁盘、光盘、或者磁带。这样额外的存储在图1中是由可移动存储设备109和不可移动存储设备110示出的。

[0018] 如在上文中所陈述的，根据一个方面，在系统存储器160中存储有程序模块140。根据一个方面，程序模块140可以包括一个或多个应用程序，一个或多个应用程序可以包括直接运行在第一操作系统上的应用程序，还可以包括适配于第二操作系统的应用程序。这里，本发明不限制应用程序的类型，例如应用程序可以包括：电子邮件和联系人应用程序、文字处理应用程序、电子表格应用程序、数据库应用程序、幻灯片展示应用程序、绘画或计算机辅助应用程序、网络浏览器应用程序等。

[0019] 在根据本发明的实施例中，程序模块140还可以包括仿真器230、第二操作系统的兼容层240，仿真器230可以运行在第一操作系统上，且通过仿真器230可以实现在第一架构处理器上的第一操作系统上仿真运行兼容层240，以便基于兼容层240来运行第二操作系统的应用程序(第二应用程序250)。应当指出，为了将第二操作系统的应用程序与第一操作系统的应用程序进行区分，可以将第二操作系统的应用程序称为“第二应用程序”，第一操作系统的应用程序称为“第一应用程序”。另外，程序模块140还可以包括调试工具260，调试工具260用于对仿真器230进行调试。

[0020] 根据一个方面，可以在包括分立电子元件的电路、包含逻辑门的封装或集成的电子芯片、利用微处理器的电路、或者在包含电子元件或微处理器的单个芯片上实践示例。例如，可以经由其中在图1中所示出的每个或许多组件可以集成在单个集成电路上的片上系统(SOC)来实践示例。根据一个方面，这样的SOC设备可以包括一个或多个处理单元、图形单元、通信单元、系统虚拟化单元、以及各种应用功能，其全部作为单个集成电路而被集成(或“烧”)到芯片基底上。当经由SOC进行操作时，可以经由在单个集成电路(芯片)上与计算设备100的其他组件集成的专用逻辑来对在本文中所描述的功能进行操作。还可以使用能够执行逻辑操作(例如AND、OR和NOT)的其他技术来实践本发明的实施例，所述其他技术包括但不限于机械、光学、流体、和量子技术。另外，可以在通用计算机内或在任何其他任何电路或系统中实践本发明的实施例。

[0021] 根据一个方面，计算设备100还可以具有一个或多个输入设备112，例如键盘、鼠标、笔、语音输入设备、触摸输入设备等。还可以包括输出设备114，例如显示器、扬声器、打印机等。前述设备是示例并且也可以使用其他设备。计算设备100可以包括允许与其他计算设备118进行通信的一个或多个通信连接116。合适的通信连接116的示例包括但不限于：RF发射机、接收机和/或收发机电路；通用串行总线(USB)、并行和/或串行端口。

[0022] 如在本文中所使用的术语计算机可读介质包括计算机存储介质。计算机存储介质可以包括以任何用于存储信息(例如，计算机可读指示、数据结构、或程序模块)的方法或技术来实现的易失性的和非易失性的、可移动的和不可移动的介质。系统存储器160、可移动

存储设备109、和不可移动存储设备110都是计算机存储介质的示例(即,存储器存储)。计算机存储介质可以包括随机存取存储器(RAM) 、只读存储器(ROM) 、电可擦只读存储器(EEPROM)、闪速存储器或其他存储器技术、CD-ROM、数字通用盘(DVD)或其他光存储、盒式磁带、磁带、磁盘存储器或其他磁存储设备、或者可用于存储信息并且可以由计算设备100访问的任何其他制品。根据一个方面,任何这样的计算机存储介质都可以是计算设备100的一部分。计算机存储介质不包括载波或其他经传播的数据信号。

[0023] 根据一个方面,通信介质是由计算机可读指令、数据结构、程序模块、或者经调制的数据信号(例如,载波或其他传输机制)中的其他数据实施的,并且包括任何信息传递介质。根据一个方面,术语“经调制的数据信号”描述了具有一个或多个特征集或者以将信息编码在信号中的方式改变的信号。作为示例而非限制,通信介质包括诸如有线网络或直接有线连接之类的有线介质,以及诸如声学、射频(RF)、红外线的、以及其他无线介质之类的无线介质。

[0024] 在根据本发明的实施例中,计算设备100被配置为执行根据本发明的仿真运行可执行文件时抓取现场的方法300。计算设备100包括一个或多个处理器、以及存储有程序指令的一个或多个可读存储介质,当程序指令被配置为由一个或多个处理器执行时,使得计算设备执行本发明实施例中的仿真运行可执行文件时抓取现场的方法300。

[0025] 根据本发明的一个实施例,计算设备100中包含用于执行本发明的仿真运行可执行文件时抓取现场的方法300的多条程序指令,这些程序指令可以指示处理器执行根据本发明的仿真运行可执行文件时抓取现场的方法300。

[0026] 图2示出了根据本发明一个实施例的计算设备100中运行有第一操作系统220的示意图。如图2所示,计算设备100中包括第一架构处理器210,并运行有第一操作系统220,第一操作系统220之上布置有应用层。

[0027] 在一个实施例中,应用层可以包括一个或多个应用程序,例如包括可以直接运行在第一操作系统220上的第一应用程序。第一架构处理器210可以支持第一操作系统220的运行,第一架构处理器210和第一操作系统220可以支持第一应用程序的运行。

[0028] 在本发明的实施例中,第一操作系统220之上的应用层中还布置有仿真器230、第二操作系统的兼容层240、一个或多个第二应用程序250。这里,第二应用程序250可以是第二操作系统的应用程序,具体可以是基于第二架构处理器和第二操作系统来运行的应用程序。基于仿真器230和兼容层240,可以实现在第一操作系统220上,运行基于第二架构处理器和第二操作系统的第二应用程序250。

[0029] 需要说明的是,兼容层240可以基于第二架构处理器来运行,无法直接基于第一架构处理器210运行,从而无法直接运行在第一架构处理器210的第一操作系统220上。仿真器230可以运行在第一架构处理器210的第一操作系统220上,且通过仿真器230可以在第一架构处理器210的第一操作系统220上仿真运行(第二操作系统的)兼容层240。

[0030] 这样,在第一架构处理器210的第一操作系统220上,通过仿真器230来仿真运行兼容层240,并基于兼容层240来运行第二操作系统的第二应用程序,从而能实现在第一架构处理器210的第一操作系统220上,运行基于第二架构处理器和第二操作系统的第二应用程序250。

[0031] 在本发明的实施例中,应用层中还布置有调试工具260,调试工具260可以直接运

行在第一操作系统220上，用于对仿真器230进行调试。

[0032] 在本发明的一个实施例中，第一架构可以为ARM架构，第二架构可以为X86架构(包含X86_64架构)等。相应地，第一架构处理器210可以为ARM处理器。

[0033] 第一操作系统220例如为Linux操作系统，第二操作系统例如为Windows操作系统。

[0034] 在一种实现方式中，第二操作系统的兼容层240可以实现为Wine(Wine is not an emulator，通过在X86架构的 Linux操作系统上实现Windows的系统函数，使得Windows应用程序在X86架构的Linux操作系统上可以正常运行的兼容层)。

[0035] 在一种实现方式中，仿真器230可以实现为box86，box86为Linux系统上的用户层X86仿真器，可以实现在非 X86架构 Linux 系统上运行 X86架构的Linux 程序。

[0036] 图3示出了根据本发明一个实施例的仿真运行可执行文件时抓取现场的方法300的流程示意图。方法300适于在计算设备(前述计算设备100)的中执行。

[0037] 如图3所示，方法300可以包括步骤310~330。

[0038] 在步骤310中，通过仿真器230，将第二应用程序250的可执行文件的第二架构指令转换为第一架构指令，并由第一架构处理器210执行该第一架构指令。这里，可执行文件例如为PE文件。

[0039] 具体地，通过调试工具260对仿真器230进行调试，在通过调试工具260对仿真器230进行调试时，通过仿真器230将第二应用程序250的可执行文件的第二架构指令进行解析、翻译，以便将可执行文件的第二架构指令转换为第一架构指令，并由第一架构处理器210执行第一架构指令。在一种实现方式中，调试工具260可以实现为gdb。

[0040] 需要说明的是，在通过仿真器230，将第二应用程序250的可执行文件的第二架构指令转换为第一架构指令，并由第一架构处理器210执行时，第一架构处理器的第一寄存器指向第二架构现场结构体。这里，第一架构处理器的第一寄存器具体可以为r0寄存器。

[0041] 在将第二应用程序250的可执行文件的第二架构指令转换为第一架构指令，并由第一架构处理器执行的过程中，如果第一架构处理器运行中断，则执行下述步骤320和330。

[0042] 在步骤320中，响应于第一架构处理器运行中断，在第一架构处理器的中断现场，获取用户输入的第一调试扩展命令，并执行该第一调试扩展命令。通过执行该第一调式扩展命令，可以自动侦测中断现场的第二架构现场结构体上下文，基于第二架构现场结构体上下文中的第二寄存器进行栈回溯，以便获取第二架构现场的调用栈。这里，调用栈中每一栈帧对应一个函数。

[0043] 需要说明的是，第二架构现场，即执行第二架构指令的现场。在一种实现方式中，第二架构指令可以为X86指令，相应地，第二架构现场结构体例如可以为x86emu_t。

[0044] 这里，由于第一架构处理器的第一寄存器指向第二架构现场结构体(x86emu_t)。因此，在基于第二架构现场结构体上下文中的第二寄存器进行栈回溯之前，可以基于第一架构处理器的第一寄存器(r0)来获取第二架构现场结构体(x86emu_t)。

[0045] 在一个实施例中，第二寄存器可以为ebp寄存器。需要说明的是，基于第二架构现场结构体可以获取第二架构现场的第二寄存器，随后，可以基于第二寄存器(ebp寄存器)来进行栈回溯，从而能够获取到第二架构现场的调用栈。

[0046] 在步骤330中，在第一架构处理器的中断现场，获取用户输入的第二调试扩展命令，并执行该第二调试扩展命令。通过执行该第二调式扩展命令，可以自动侦测中断现场的

第二架构现场结构体(x86emu_t)上下文,从而可以从第二架构现场结构体(x86emu_t)上下文中获取到第二架构现场的寄存器信息。

[0047] 应当指出,在上述步骤320中获取到的第二架构现场的调用栈,只有地址,没有符号信息。

[0048] 图4示出了根据本发明一个实施例的仿真运行可执行文件时抓取现场的方法的时序图。

[0049] 如图4所示,可以通过以下方法来获取带符号的调用栈。

[0050] 在执行本发明的方法300之前,预先在兼容层240中置入符号信息。

[0051] 在通过仿真器230,将第二应用程序250的可执行文件的第二架构指令转换为第一架构指令并执行之前,首先通过兼容层240加载第二应用程序250的可执行文件。

[0052] 其中,兼容层240在加载第二应用程序250的可执行文件时,可以收集兼容层240中预先置入的符号信息。随后,兼容层240可以将符号信息发送至仿真器230。这里,当仿真器230需要识别地址的符号时,仿真器230可以通过地址解析符号信息。

[0053] 仿真器230在接收到兼容层240发送的符号信息之后,可以通过仿真器230执行前述步骤310~320,来获取第二架构现场的调用栈。这样,在获取到第二架构现场的调用栈之后,仿真器230可以将符号信息与调用栈进行匹配,从而能得到第二架构现场的带符号调用栈。这样,便能实现从第一架构处理器的中断现场,获取到第二架构现场的带符号调用栈。

[0054] 在一种实现方式中,兼容层240可以利用build_module函数来加载可执行文件(PE文件),在加载第二应用程序250的可执行文件时,通过解析兼容层240的导出表符号信息并进行处理,来获取符号信息。具体地,可以通过在兼容层240中插入相应代码来收集获取相关符号信息,并且,不会干扰兼容层240的正常运行流程。

[0055] 在一个实施例中,如前文所述,在步骤310中,是在通过调试工具260对仿真器230进行调试时,通过仿真器230将第二应用程序250的可执行文件的第二架构指令转换为第一架构指令,并由第一架构处理器执行第一架构指令。需要说明的是,调试工具260与仿真器230通信连接。

[0056] 在该实施例中,如图4所示,在通过调试工具260对仿真器230进行调试时,首先,调试工具260会向仿真器230请求获取第二架构现场(包括第二架构现场的调用栈以及寄存器信息)。

[0057] 随后,仿真器230(在获取到符号信息之后)接收调试工具260发送的获取第二架构现场的请求,并响应于调试工具260发送的获取第二架构现场的请求,通过执行前述步骤310~320,来获取第二架构现场的调用栈。在获取到第二架构现场的调用栈之后,仿真器230可以将符号信息与调用栈进行匹配,从而能得到第二架构现场的带符号调用栈。并且,仿真器230通过执行前述步骤330,来获取第二架构现场的寄存器信息。

[0058] 最后,仿真器230在获取到第二架构现场的带符号调用栈以及寄存器信息之后,可以将第二架构现场的带符号调用栈、第二架构现场的寄存器信息返回至调试工具260,从而实现向调试工具260返回第二架构现场。

[0059] 需要说明的是,基于第二架构现场结构体上下文中的第二寄存器(ebp寄存器)进行栈回溯,以便获取第二架构现场的调用栈,具体方法如下。

[0060] 基于当前ebp寄存器,获取返回地址。基于该返回地址回溯到前一函数,并获取前

一函数对应的上层ebp寄存器，上层ebp寄存器保存有前一栈帧。

[0061] 基于上层ebp寄存器，获取新的返回地址。基于新的返回地址回溯到新的前一函数，并获取新的前一函数对应的新的上层ebp寄存器，直至获取到完整调用栈。这里，新的上层ebp寄存器保存有新的前一栈帧。

[0062] 图5示出了根据本发明一个实施例的调用栈的分布示意图。

[0063] 需要说明的是，在程序运行过程中，例如，在第一架构处理器执行可执行文件的第一架构指令的过程中，发生函数调用时，会生成一个调用栈。如图5所示，调用栈中存储有调用者函数的参数(例如参数1，参数2…参数N)、调用者函数的返回地址、寄存器ebp的值、局部变量等数据。每当发生函数调用，会自动将函数添加到调用栈中并执行函数，在执行完成该函数后，调用栈会自动移除该函数。

[0064] 第二寄存器(ebp寄存器)的指针指向调用栈中最上面栈帧。当前第二寄存器中保存有当前函数的栈帧。

[0065] 基于第二架构现场结构体上下文中的第二寄存器(ebp寄存器)进行栈回溯，以便获取第二架构现场的调用栈，具体方法如下。

[0066] 首先，可以基于当前ebp寄存器，可以从前一函数的前一栈帧中(顶部)获取返回地址，即获取前一函数的返回地址，前一函数也即当前函数的调用者函数。

[0067] 接着，可以基于前一函数的返回地址回溯到前一函数(当前函数的调用者函数)，并可以获取前一函数对应的上层ebp寄存器，这里，上层ebp寄存器中保存有前一函数对应的前一栈帧。

[0068] 接下来，可以判断是否获取到完整调用栈，也即判断在前一栈帧之前是否还包括更多栈帧。如果没有获取到完整调用栈(还包括更多栈帧)，此时，可以将前一函数看作是新的当前函数，将上层ebp寄存器看作是新的当前ebp寄存器。并继续执行前述步骤。

[0069] 具体地，基于上层ebp寄存器，获取(新的当前函数的前一函数的)新的返回地址，基于新的返回地址可以继续回溯到新的当前函数的前一函数(新的前一函数)，并可以获取到新的前一函数对应的新的上层ebp寄存器。这里，新的上层ebp寄存器中保存有新的前一函数对应的新的前一栈帧。

[0070] 如果没有获取到完整调用栈，则继续重复执行以上步骤，直至获取到完整调用栈。

[0071] 在一个实施例中，在通过仿真器230将第二应用程序250的可执行文件第二架构指令转换为第一架构指令，并由第一架构处理器执行第一架构指令的过程中，当发生函数调用时，仿真器230可以拦截相关函数，并将函数封装后转发到第一架构处理器上的第一操作系统调用。

[0072] 在一种实现方式中，仿真器230可以实现为box86，可以利用box86的库包装技术，来拦截执行第一架构指令时调用的函数，并将该函数封装后转发到ARM系统调用。例如，仿真器230可以将getrandom函数封装为my_getrandom。

[0073] 这样，在对兼容层240的代码改动后(通过插入代码来收集符号信息)，即使兼容层240运行在X86环境，没有box86接管的情况下，也可以确保兼容层240正常运行，不会出现错误。

[0074] 根据本发明的仿真运行可执行文件时抓取现场的方法300，在布置有第一架构处理器及第一操作系统的计算设备中，基于仿真器仿真运行兼容层、以执行第二操作系统的

第二应用程序时，通过仿真器将第二应用程序的可执行文件的第二架构指令转换为第一架构指令，并由第一架构处理器执行，如果出现运行中断，则在第一架构处理器的中断现场，获取输入的第一调试扩展命令并执行，以侦测中断现场的第二架构现场结构体上下文，基于其中的第二寄存器进行栈回溯，以便获取第二架构现场的调用栈。并且，在第一架构处理器的中断现场，获取输入的第二调试扩展命令并执行，以侦测中断现场的第二架构现场结构体上下文，获取第二架构现场的寄存器信息。这样，本发明能实现在第一架构环境下执行可执行文件时，可以实现在第一架构处理器的中断现场，简单方便地直接查看第二架构现场的调用栈和寄存器信息，从而能直观地对应到第二架构指令的错误现场，并且，可以基于第二架构现场的调用栈和寄存器信息来分析可执行文件在当前运行环境下(第一架构的第一操作系统)的执行逻辑，并可以横向对比可执行文件在第二架构的第二操作系统上的运行流程，这样有助于发现导致运行错误的原因，高效解决问题，最终实现在第一架构的第一操作系统上正常运行可执行文件。

[0075] 进一步地，通过在兼容层预先置入符号信息，在兼容层加载第二应用程序的可执行文件时通过截取符号信息，并将符号信息发送至仿真器。这样，仿真器在获取到第二架构现场的调用栈之后，可以将符号信息与调用栈进行匹配，从而能得到带符号调用栈。这样，能实现从第一架构处理器的中断现场，获取到第二架构现场的带符号调用栈。

[0076] 这里描述的各种技术可结合硬件或软件，或者它们的组合一起实现。从而，本发明的方法和设备，或者本发明的方法和设备的某些方面或部分可采取嵌入有形媒介，例如可移动硬盘、U盘、软盘、CD-ROM或者其它任意机器可读的存储介质中的程序代码(即指令)的形式，其中当程序被载入诸如计算机之类的机器，并被所述机器执行时，所述机器变成实践本发明的设备。

[0077] 在程序代码在可编程计算机上执行的情况下，移动终端一般包括处理器、处理器可读的存储介质(包括易失性和非易失性存储器和/或存储元件)，至少一个输入装置，和至少一个输出装置。其中，存储器被配置用于存储程序代码；处理器被配置用于根据该存储器中存储的所述程序代码中的指令，执行本发明的仿真运行可执行文件时抓取现场的方法。

[0078] 以示例而非限制的方式，可读介质包括可读存储介质和通信介质。可读存储介质存储诸如计算机可读指令、数据结构、程序模块或其它数据等信息。通信介质一般以诸如载波或其它传输机制等已调制数据信号来体现计算机可读指令、数据结构、程序模块或其它数据，并且包括任何信息传递介质。以上的任一种的组合也包括在可读介质的范围之内。

[0079] 在此处所提供的说明书中，算法和显示不与任何特定计算机、虚拟系统或者其它设备固有相关。各种通用系统也可以与本发明的示例一起使用。根据上面的描述，构造这类系统所要求的结构是显而易见的。此外，本发明也不针对任何特定编程语言。应当明白，可以利用各种编程语言实现在此描述的本发明的内容，并且上面对特定语言所做的描述是为了披露本发明的最佳实施方式。

[0080] 在此处所提供的说明书中，说明了大量具体细节。然而，能够理解，本发明的实施例可以在没有这些具体细节的情况下被实践。在一些实例中，并未详细示出公知的方法、结构和技术，以便不模糊对本说明书的理解。

[0081] 类似地，应当理解，为了精简本公开并帮助理解各个发明方面中的一个或多个，在上面对本发明的示例性实施例的描述中，本发明的各个特征有时被一起分组到单个实施

例、图、或者对其的描述中。

[0082] 本领域那些技术人员应当理解在本文所公开的示例中的设备的模块或单元或组件可以布置在如该实施例中所描述的设备中，或者可替换地可以定位在与该示例中的设备不同的一个或多个设备中。前述示例中的模块可以组合为一个模块或者此外可以分成多个子模块。

[0083] 本领域那些技术人员可以理解，可以对实施例中的设备中的模块进行自适应性地改变并且把它们设置在与该实施例不同的一个或多个设备中。可以把实施例中的模块或单元或组件组合成一个模块或单元或组件，以及此外可以把它们分成多个子模块或子单元或子组件。

[0084] 此外，本领域的技术人员能够理解，尽管在此所述的一些实施例包括其它实施例中所包括的某些特征而不是其它特征，但是不同实施例的特征的组合意味着处于本发明的范围之内并且形成不同的实施例。

[0085] 此外，所述实施例中的一些在此被描述成可以由计算机系统的处理器或者由执行所述功能的其它装置实施的方法或方法元素的组合。因此，具有用于实施所述方法或方法元素的必要指令的处理器形成用于实施该方法或方法元素的装置。此外，装置实施例的在此所述的元素是如下装置的例子：该装置用于实施由为了实施该发明的目的的元素所执行的功能。

[0086] 如在此所使用的那样，除非另行规定，使用序数词“第一”、“第二”、“第三”等等来描述普通对象仅仅表示涉及类似对象的不同实例，并且并不意图暗示这样被描述的对象必须具有时间上、空间上、排序方面或者以任意其它方式的给定顺序。

[0087] 尽管根据有限数量的实施例描述了本发明，但是受益于上面的描述，本技术领域内的技术人员明白，在由此描述的本发明的范围内，可以设想其它实施例。此外，应当注意，本说明书中使用的语言主要是为了可读性和教导的目的而选择的，而不是为了解释或者限定本发明的主题而选择的。

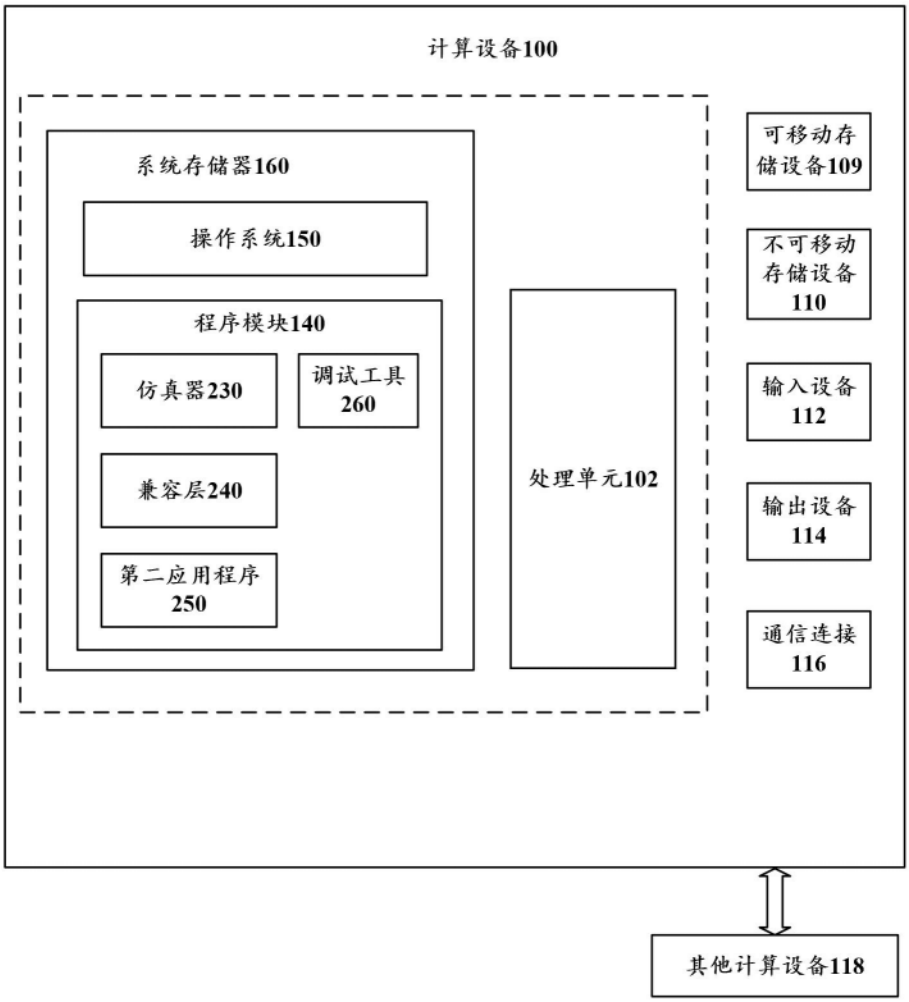

计算设备100
系统存储器160
操作系统150
程序模块140
仿真器230
调试工具260
兼容层240
第二应用程序250
处理单元102
可移动存储设备109
不可移动存储设备110
输入设备112
输出设备114
通信连接116
其他计算设备118

图 1

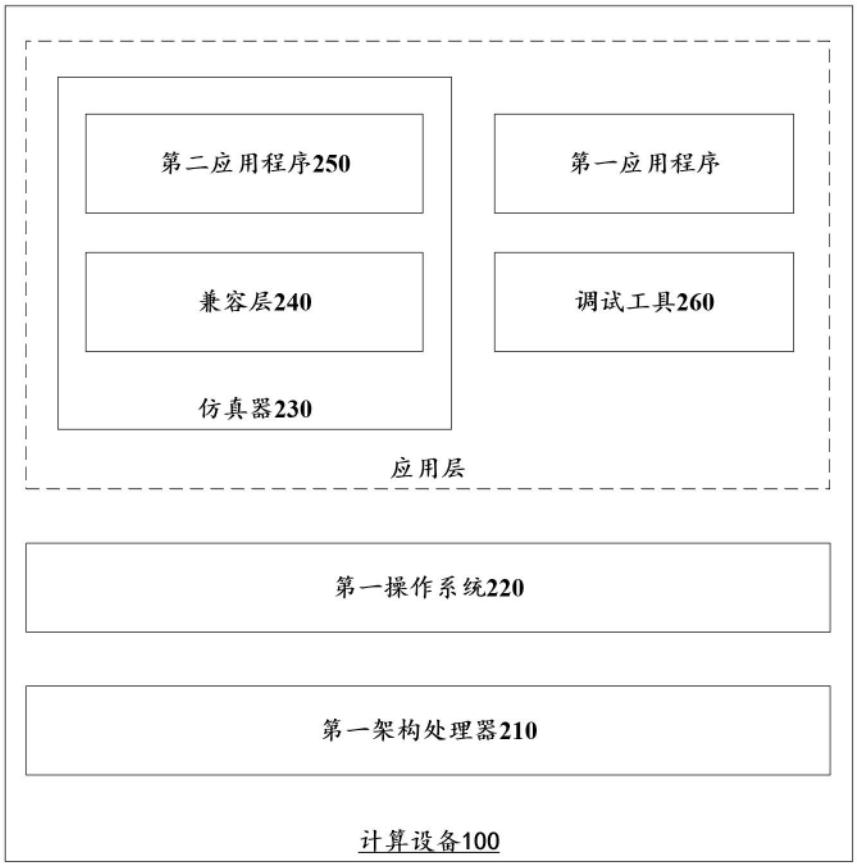

第二应用程序250
第一应用程序
兼容层240
调试工具260
仿真器230
应用层
第一操作系统220
第一架构处理器210
计算设备100

图 2

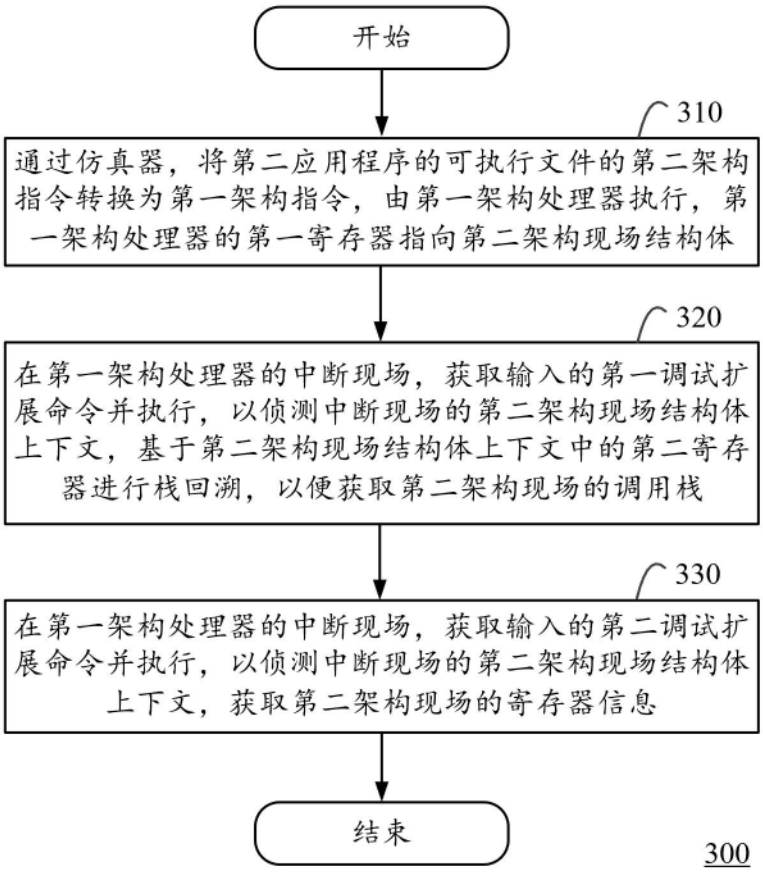

开始
310
通过仿真器，将第二应用程序的可执行文件的第二架构指令转换为第一架构指令，由第一架构处理器执行，第一架构处理器的第一寄存器指向第二架构现场结构体
320
在第一架构处理器的中断现场，获取输入的第一调试扩展命令并执行，以侦测中断现场的第二架构现场结构体上下文，基于第二架构现场结构体上下文中的第二寄存器进行栈回溯，以便获取第二架构现场的调用栈
330
在第一架构处理器的中断现场，获取输入的第二调试扩展命令并执行，以侦测中断现场的第二架构现场结构体上下文，获取第二架构现场的寄存器信息
结束
300

图 3

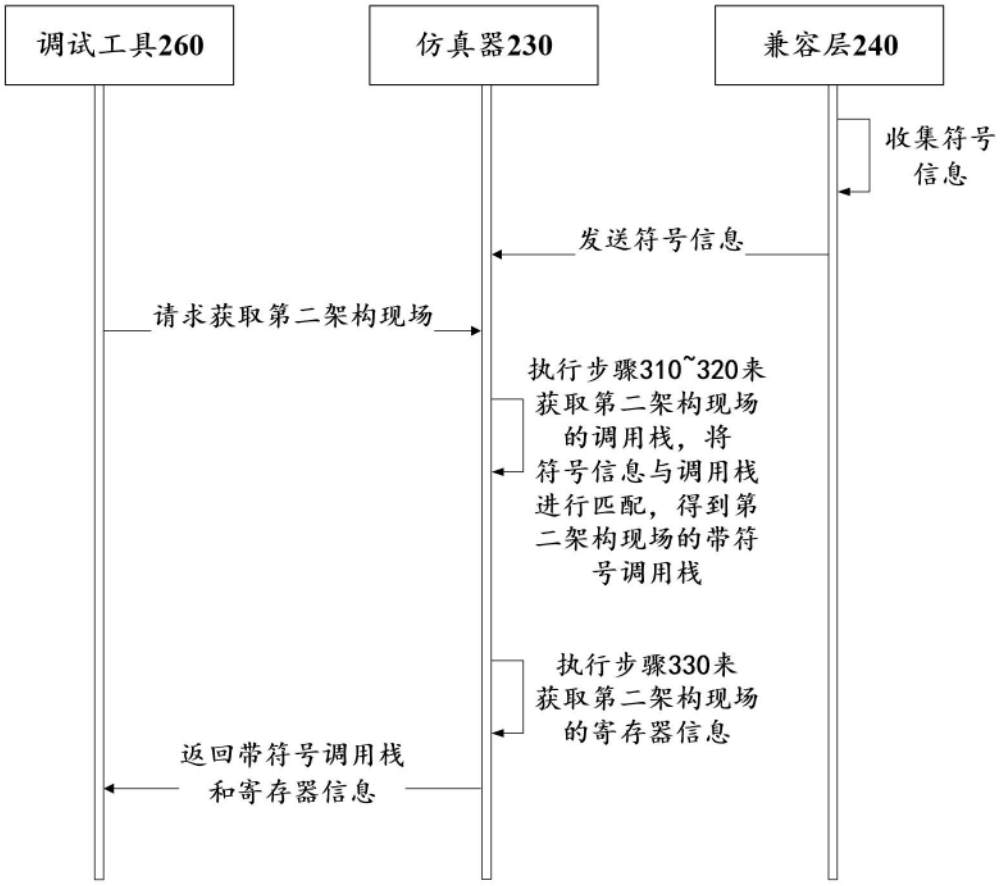

调试工具260
仿真器230
兼容层240
收集符号信息
发送符号信息
请求获取第二架构现场
执行步骤310~320来获取第二架构现场的调用栈，将符号信息与调用栈进行匹配，得到第二架构现场的带符号调用栈
执行步骤330来获取第二架构现场的寄存器信息
返回带符号调用栈和寄存器信息

图 4

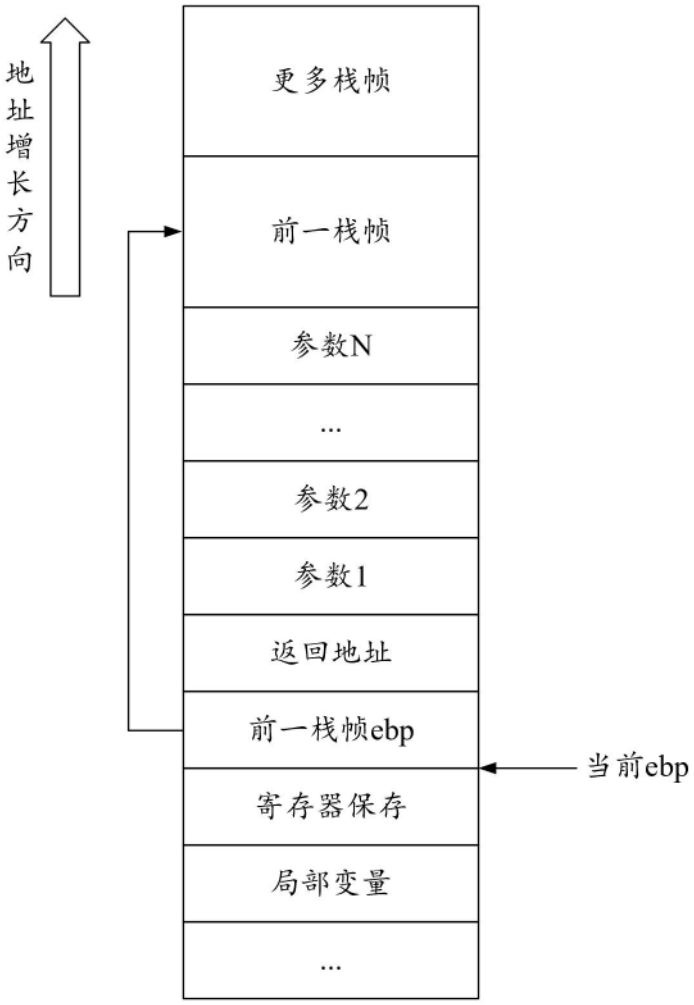


图 5