(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 115952007 B
(45)授权公告日 2023.06.16

(21)申请号 202310246391.4

(22)申请日 2023.03.09

(65)同一申请的已公布的文献号
申请公布号 CN 115952007 A

(43)申请公布日 2023.04.11

(73)专利权人 杭州银行股份有限公司
地址 310000 浙江省杭州市下城区庆春路46号

(72)发明人 胡飞华 潘嘉钦 马奇 李凤超 陈志荣 王庆峰

(74)专利代理机构 北京思睿峰知识产权代理有限公司 11396
专利代理师 谢建云 赵爱军

(51)Int.Cl.
G06F 9/50(2006.01)
G06F 8/61(2018.01)
G06F 11/14(2006.01)

(56)对比文件
WO 2017140216 A1,2017.08.24
US 2020285488 A1,2020.09.10
CN 111130852 A,2020.05.08
沈磊;何娇;胡昊.分布式环境应用系统运行状态监控方法研究.中国环境管理.2016,(02),全文.

审查员 曹宁

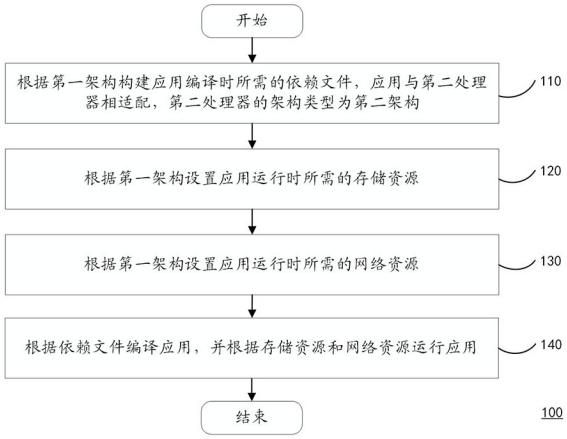
权利要求书2页 说明书12页 附图2页

(54)发明名称

一种应用运行方法、计算设备及存储介质

(57)摘要

本发明涉及云计算领域,特别涉及一种应用运行方法、计算设备及存储介质,计算设备中包括第一处理器,第一处理器的架构类型为第一架构,方法包括步骤:根据第一架构构建应用编译时所需的依赖文件,应用与第二处理器相适配,第二处理器的架构类型为第二架构;根据第一架构设置应用运行时所需的存储资源;根据第一架构设置应用运行时所需的网络资源;根据依赖文件编译应用,并根据存储资源和网络资源运行应用。本发明中通过对应用编译所需文件进行构建,实现在未适配相应处理器架构的计算设备中能够启动应用,并且配置所需的存储资源和网络资源,实现应用在未适配相应处理器架构的计算设备中运行。



CN 115952007 B

1.一种应用运行方法,适于在计算设备中执行,所述计算设备中包括第一处理器,所述第一处理器的架构类型为第一架构,所述方法包括步骤:

根据所述第一架构构建应用编译时所需的依赖文件,所述应用与第二处理器相适配,所述第二处理器的架构类型为第二架构;

根据所述第一架构设置所述应用运行时所需的存储资源;

根据所述第一架构设置所述应用运行时所需的网络资源;

根据所述依赖文件编译所述应用,并根据所述存储资源和网络资源运行所述应用;

其中,所述根据所述第一架构构建应用编译时所需的依赖文件,包括:

获取所述应用编译时所需的一个或多个库文件;

根据测试应用或库文件的声明文件判断每个库文件是否适配第一架构;

根据所述第一架构定义结构体;

根据所述结构体和一个或多个库文件生成所述依赖文件,其中,对不适配第一架构的库文件修改路径,使用本地的库文件生成依赖文件;所述根据所述第一架构设置所述应用运行时所需的存储资源,包括:

根据所述第一架构构建依赖库文件;

根据所述依赖库文件生成适于由所述第一架构的第一处理器运行的安装包;

根据所述安装包在所述计算设备中构建存储节点,并根据所述存储节点设置所述应用运行时所需的存储资源;

所述根据所述第一架构构建依赖库文件,包括:

根据所述第一架构设置存储映射参数和存储指针参数;

根据所述存储映射参数和所述存储指针参数构建依赖库文件;

所述方法还包括:

设置默认后端镜像和控制器镜像;

根据所述默认后端镜像和所述控制器镜像设置负载均衡服务;

根据所述负载均衡服务确定应用的运行信息,所述运行信息包括流量流转过程和调用服务过程;

通过应用的运行信息确定应用是否运行正常;

设置默认后端镜像时,获取未适配的默认后端镜像文件以便适配第一架构;

编译未适配的默认后端镜像文件;

修改编译后的默认后端镜像文件的文件标识;

设置控制器镜像时,获取未适配的控制器镜像文件并进行测试;

根据报错信息确定缺失函数;

在未适配的控制器镜像文件中构造缺失函数以便适配第一架构。

2.如权利要求1所述的方法,其中,所述根据所述第一架构设置所述应用运行时所需的网络资源包括:

获取第一镜像文件,所述第一镜像文件适于由所述第一架构的第一处理器运行;

如果安装网络组件的所需镜像未适配第一架构,则根据所述第一架构构建第二镜像文件;

根据所述第一镜像文件和所述第二镜像文件安装网络组件;

根据所述网络组件设置所述应用运行时所需的网络资源。

3.如权利要求2所述的方法,其中,根据所述第一架构构建第二镜像文件,包括:

获取所述网络组件所需的一个或多个库文件;

根据一个或多个库文件,以及所述第一架构构建第二镜像文件。

4.如权利要求1所述的方法,其中,所述计算设备与一个或多个其他计算设备通信连接,所述方法还包括:

在其他计算设备中构建一个或多个备份存储资源;

当所述计算设备所提供的存储资源不可用时,所述应用根据所述备份存储资源运行。

5.一种计算设备,包括:

一个或多个处理器;

存储器;以及

一个或多个程序,其中,一个或多个程序存储在存储器中并被配置为由一个或多个处理器执行,一个或多个程序包括用于执行根据权利要求1-4中任一项所述的方法的指令。

6.一种存储一个或多个程序的计算机可读存储介质,所述一个或多个程序包括指令,所述指令当由计算设备执行时,使得所述计算设备执行根据权利要求1-4中任一项所述的方法。

一种应用运行方法、计算设备及存储介质

技术领域

[0001] 本发明涉及云计算领域，特别涉及一种应用运行方法、计算设备及存储介质。

背景技术

[0002] 云计算领域近年来迅速发展，云计算是指IT基础设施的交付和使用模式，即通过网络以按需、易扩展的方式获取所需资源。广义上则指服务的交付和使用模式，通过网络以按需、易扩展的方式获取所需服务。提供资源的网络被形象地比喻成“云”，其计算能力通常是由分布式的大规模集群和虚拟化技术提供的。

[0003] 在搭建“云”，使用云中的节点进行提供服务时，由于节点与节点之间所采用的处理器的架构不同，在部署应用时提供服务时会出现适配性的问题。若应用对某个处理器架构的节点上不能适配运行，那么该节点就不能部署应用提供服务，这样就大幅限制了云集群提供服务的性能。

[0004] 为此，需要一种新的应用运行方法。

发明内容

[0005] 为此，本发明提供一种应用运行方法，以力图解决或者至少缓解上面存在的问题。

[0006] 根据本发明的一个方面，提供一种应用运行方法，适于在计算设备中执行，计算设备中包括第一处理器，第一处理器的架构类型为第一架构，方法包括步骤：根据第一架构构建应用编译时所需的依赖文件，应用与第二处理器相适配，第二处理器的架构类型为第二架构；根据第一架构设置应用运行时所需的存储资源；根据第一架构设置应用运行时所需的网络资源；根据依赖文件编译应用，并根据存储资源和网络资源运行应用。

[0007] 可选地，在根据本发明的方法中，根据第一架构构建应用编译时所需的依赖文件，包括：获取应用编译时所需的一个或多个库文件；根据第一架构定义结构体；根据结构体和一个或多个库文件生成依赖文件。

[0008] 可选地，在根据本发明的方法中，根据第一架构设置应用运行时所需的存储资源，包括：根据第一架构构建依赖库文件；根据依赖库文件生成适于由第一架构的第一处理器运行的安装包；根据安装包在计算设备中构建存储节点，并根据存储节点设置应用运行时所需的存储资源。

[0009] 可选地，在根据本发明的方法中，根据安装信息从资源服务器中获取第一引导程序包括步骤：根据安装信息从资源服务器获取信息文件；根据所获取的信息文件从资源服务器获取第一引导程序。

[0010] 可选地，在根据本发明的方法中，根据第一架构构建依赖库文件，包括：根据第一架构设置存储映射参数和存储指针参数；根据存储映射参数和存储指针参数构建依赖库文件。

[0011] 可选地，在根据本发明的方法中，根据第一架构设置应用运行时所需的网络资源包括：获取第一镜像文件，第一镜像文件适于由第一架构的第一处理器运行；如果安装网络

组件的所需镜像未适配第一架构，则根据第一架构构建第二镜像文件；根据第一镜像文件和第二镜像文件安装网络组件；根据网络组件设置应用运行时所需的网络资源。

[0012] 可选地，在根据本发明的方法中，根据第一架构构建第二镜像文件，包括：获取网络组件所需的一个或多个库文件；根据一个或多个库文件，以及第一架构构建第二镜像文件。

[0013] 可选地，在根据本发明的方法中，计算设备与一个或多个其他计算设备通信连接，方法还包括：在其他计算设备中构建一个或多个备份存储资源；当计算设备所提供的存储资源不可用时，应用根据备份存储资源运行。

[0014] 可选地，在根据本发明的方法中，方法还包括：设置默认后端镜像和控制器镜像；根据默认后端镜像和控制器镜像设置负载均衡服务；根据负载均衡服务确定应用是否运行正常。

[0015] 根据本发明的另一个方面，提供了一种计算设备，包括：一个或多个处理器；存储器；以及一个或多个程序，其中，一个或多个程序存储在存储器中并被配置为由一个或多个处理器执行，一个或多个程序包括用于执行根据本发明的应用运行方法的指令。

[0016] 根据本发明的再一个方面，提供了一种存储一个或多个程序的计算机可读存储介质，一个或多个程序包括指令，该指令当由计算设备执行时，使得计算设备执行根据本发明的应用运行方法。

[0017] 本发明公开了一种应用运行方法，适于在计算设备中执行，计算设备中包括第一处理器，第一处理器的架构类型为第一架构，方法包括步骤：根据第一架构构建应用编译时所需的依赖文件，应用与第二处理器相适配，第二处理器的架构类型为第二架构；根据第一架构设置应用运行时所需的存储资源；根据第一架构设置应用运行时所需的网络资源；根据依赖文件编译应用，并根据存储资源和网络资源运行应用。本发明中通过对应用编译所需文件进行构建，实现在未适配相应处理器架构的计算设备中能够启动应用，并且配置所需的存储资源和网络资源，实现应用在未适配相应处理器架构的计算设备中运行。

附图说明

[0018] 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，本发明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。

[0019] 图1示出了根据本发明一个示范性实施例的应用运行方法100的流程示意图；

[0020] 图2示出了根据本发明一个示范性实施例的计算设备200的结构框图。

具体实施方式

[0021] 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。相同的附图标记通常指代相同的部件或元素。

[0022] 本发明中的一种应用运行方法适于在计算设备中执行。图2示出了根据本发明一个示范性实施例的计算设备的结构框图。

[0023] 在基本配置中，计算设备200包括至少一个处理单元220和系统存储器210。根据一个方面，取决于计算设备的配置和类型，系统存储器210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210包括操作系统211。

[0024] 根据一个方面，操作系统211，例如，适合于控制计算设备200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2中通过在虚线215内的那些组件示出了该基本配置。根据一个方面，计算设备200具有额外的特征或功能。例如，根据一个方面，计算设备200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。

[0025] 如在上文中所陈述的，根据一个方面，在系统存储器210中存储程序模块212。根据一个方面，程序模块212可包括一个或多个应用程序，本发明不限制应用程序的类型，例如应用还包括：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。

[0026] 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2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200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。

[0027] 根据一个方面，计算设备200还可以具有一个或多个输入设备231，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232，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200可以包括允许与其他计算设备240进行通信的一个或多个通信连接233。合适的通信连接233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。计算设备200可通过通信连接233与其他计算设备240通信连接。

[0028] 本发明实施方式还提供一种非暂态可读存储介质，存储有指令，所述指令用于使所述计算设备执行根据本发明实施方式的方法。本实施例的可读介质包括永久性和非永久性、可移动和非可移动介质，可以由任何方法或技术来实现信息存储。信息可以是计算机可读指令、数据结构、程序的模块或其他数据。可读存储介质的例子包括但不限于：相变内存(PRAM)、静态随机存取存储器(SRAM)、动态随机存取存储器(DRAM)、其他类型的随机存取存储器(RAM)、只读存储器(ROM)、电可擦除可编程只读存储器(EEPROM)、快闪记忆体或其他内存技术、只读光盘只读存储器(CD-ROM)、数字多功能光盘(DVD)或其他光学存储、磁盒式磁带，磁带磁盘存储或其他磁性存储设备或任何其他非暂态可读存储介质。

[0029] 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。

[0030] 需要说明的是，尽管上述计算设备仅示出了处理单元220、系统存储器210、输入设备231、输出设备232、以及通信连接233，但是在具体实施过程中，该设备还可以包括实现正常运行所必需的其他组件。此外，本领域的技术人员可以理解的是，上述设备中也可以仅包含实现本说明书实施例方案所必需的组件，而不必包含图中所示的全部组件。

[0031] 图1示出了根据本发明一个示范性实施例的应用运行方法100的流程示意图。如图1所示，首先执行步骤110，根据第一架构构建应用编译时所需的依赖文件，应用与第二处理器相适配，第二处理器的架构类型为第二架构。

[0032] 根据本发明的一个实施例，应用可实现为任意一种类型的应用，本发明对应用的具体类型不做限制。应用适配了第二处理器，第二处理器的架构类型为第二架构，应用未适配架构类型为第一架构的第一处理器。应用编译时会需要一个或多个依赖文件，本发明对应用所需依赖文件的数量及具体文件类型不做限制。

[0033] 根据本发明的一个实施例，第一架构可具体实现为龙架构(LoongArch)等，第二架构可具体实现为x86架构等，本发明对第一架构或第二架构的具体类型不做限制。

[0034] 根据第一架构构建应用编译时所需的依赖文件时，获取应用编译时所需的一个或多个库文件；根据第一架构定义结构体；再根据结构体和一个或多个库文件生成依赖文件。

[0035] 根据本发明的一个实施例，所需要的依赖文件可实现为go环境，在搭建应用所需要的依赖环境时，对依赖环境的一个或多个库文件是否适配第一架构进行判断。根据本发明的一个实施例，可预先对文件(文件包括库文件、依赖库文件等)是否适配相应架构进行判断，具体的：可通过预设的测试应用对文件进行测试，以便判断依赖文件是否适配相应架构。测试应用用于对文件是否适配架构进行判断，本发明对测试应用的具体类型及所使用的测试方法不做限制。还可通过文件的声明文件判断该文件是否适配相应架构。声明文件中记载了文件所适配的一种或多种架构类型。

[0036] 若库文件未适配第一架构的第一处理器(如gopsutil库)，则需要重新生成依赖文件。具体的，首先获取go环境未适配的一个或多个库文件，具体可通过执行如下代码实现：

[0037] go mod download。

[0038] 随后，由于go环境本身依赖的一个或多个库文件(如：gopsutil库)未适配第一架构的第一处理器，因此，将go环境所需要的库文件的路径进行修改，使用本地可用的库文件，具体可通过执行如下代码实现：

[0039] cd/root/go/pkg/mod/github.com/shirou/gopsutil@v0.0.0-20190323131628-2cbc9195c892/

[0040] GOOS="linux"

[0041] GOARCH="loong64"。

[0042] 根据本发明的一个实施例，对文件(文件包括库文件等)的路径进行修改适配相应

架构时,可通过export或者环境变量的方式去定义编译路径,修改后的路径包括文件的编译路径和该库文件的文件名,以便根据该路径使用本地库文件。

[0043] 随后,根据第一架构定义结构体,再根据结构体和一个或多个库文件生成依赖文件。根据本发明的一个实施例,根据第一架构定义结构体时,可通过cgo工具定义结构体,具体的可执行如下代码实现:

[0044] DIRS="cpu disk docker host load mem net process"

[0045] for DIR in $DIRS

[0046] do

[0047] if [-e ${DIR}/types_${GOOS}.go];then

[0048] echo "// +build $GOOS">${DIR}/${DIR}_${GOOS}_${GOARCH}.go

[0049] echo "// +build $GOARCH">>${DIR}/${DIR}_${GOOS}_${GOARCH}.go

[0050] go tool cgo -godefs ${DIR}/types_${GOOS}.go

[0051] >>${DIR}/${DIR}_${GOOS}_${GOARCH}.go

[0052] fi

[0053] done。

[0054] 随后,执行步骤120,根据第一架构设置应用运行时所需的存储资源;具体的:根据第一架构构建依赖库文件;根据依赖库文件生成适于由第一架构的第一处理器运行的安装包;根据安装包在计算设备中构建存储节点,并根据存储节点设置应用运行时所需的存储资源。

[0055] 根据第一架构构建依赖库文件时,先根据第一架构设置存储映射参数和存储指针参数,再根据存储映射参数和存储指针参数构建依赖库文件。

[0056] 根据本发明的一个实施例,根据第一架构设置存储映射参数时,存储映射参数可具体实现为maxMapSize变量,maxMapSize表示依赖库文件所支持的最大内存映射(mmap)大小,具体可根据如下代码实现设置存储映射参数:

[0057] const maxMapSize = 0xFFFFFFFFFFFF // 256TB。

[0058] 根据第一架构设置存储指针参数时,存储指针参数可具体实现为maxAllocSize,maxAllocSize是依赖库文件中创建数组指针使用的大小。

[0059] 根据本发明的一个实施例,对生成安装包所需的依赖库文件进行判断是否适配第一架构。本发明对依赖库文件的数量及具体类型不做限制。若未适配第一架构,则需要重新构建依赖库文件。根据本发明的一个实施例,所构建的依赖库文件可实现为bolt库,bolt库不适配第一架构的第一处理器,因此需要重新构建,具体的可通过自定义依赖库文件的参数实现重新构建。bolt库可不需要完整的数据库服务器,其提供一个简单、快速和可靠的数据库。

[0060] 根据本发明的一个实施例,根据存储映射参数和存储指针参数构建依赖库文件时,可具体通过执行如下代码实现:

[0061] cat>vendor/github.com/boltdb/bolt/bolt_loong64.go<<'EOF'

[0062] //go:build loong64

[0063] // +build loong64

[0064] package bolt

[0065] var brokenUnaligned = false

[0066] EOF。

[0067] 所构建的依赖库文件适配了第一架构的第一处理器,根据该依赖库文件生成的安装包,适于由第一架构的第一处理器运行。

[0068] 根据本发明的一个实施例,该安装包可实现为glusterfs安装包。随后根据该安装包在计算设备中构建glusterfs存储节点;具体可通过执行如下代码实现:

[0069] Rpm-ivh

[0070] glusterfs-8.5-0.0.a.ky10.a.ky10.loongarch64.rpm

[0071] libglusterfs0-8.5-0.0.a.ky10.a.ky10.loongarch64.rpm

[0072] libgfrpc0-8.5-0.0.a.ky10.a.ky10.loongarch64.rpm

[0073] libgfxdr0-8.5-0.0.a.ky10.a.ky10.loongarch64.rpm

[0074] rpm -ivh glusterfs-client-xlators-8.5-0.0.a.ky10.a.ky10.loongarch64.rpm

[0075] rpm -ivh glusterfs-fuse-8.5-0.0.a.ky10.a.ky10.loongarch64.rpm

[0076] cp $bashpath/heketi-10.4-loong64/heketi /usr/bin/

[0077] cp $bashpath/heketi-10.4-loong64/heketi-cli /usr/bin/

[0078] cp $bashpath/heketi-10.4-loong64/heketi.service /usr/lib/systemd/system

[0079] useradd -r heketi

[0080] mkdir -p /etc/heketi&&chown -R heketi:heketi /etc/heketi

[0081] mkdir -p /var/lib/heketi&&chown -R heketi:heketi /var/lib/heketi

[0082] cp $bashpath/heketi-10.4-loong64/heketi.json /etc/heketi/heketi.json

[0083] cp $bashpath/heketi-10.4-loong64/topology.json /etc/heketi/topology.json。

[0084] 根据本发明的一个实施例,构建的存储节点包括主节点、备节点和仲裁节点。本发明对主节点和备节点的数目不做限制。主节点和备节点存储数据。在主节点正常运行时,主节点正常提供对所存储数据的访问并能够存储数据。当主节点出现故障不能正常运行时,由仲裁节点决定哪一个备节点提升为主节点,以便继续提供存储数据和访问数据的服务。

[0085] 本发明通过在多个计算设备中构建多个存储节点,并且存储节点包括多种类型,有效降低甚至消除物理机宕机对业务造成的影响,实现故障隔离,提高存储服务的可靠性。

[0086] 根据存储节点设置应用运行所需的存储资源时,设置配置文件,根据配置文件设置应用运行所需的存储资源。根据本发明的一个实施例,配置文件可具体实现为yaml文件,本发明对配置文件的具体类型不作限制。设置配置文件可具体通过执行如下代码实现:

[0087] apiVersion: storage.k8s.io/v1

[0088] kind: StorageClass

[0089] metadata:

[0090] annotations:

[0091] storageLimit: "50"

[0092] type: GlusterFS

[0093] name: default

[0094] parameters:

[0095] clusterid: 2d0e7e7353414abd4f600cd36813be9f

[0096] restauthenabled: "true"

[0097] resturl: http://168.68.29.72:8080

[0098] restuser: admin

[0099] secretName: glusterfs-admin-secret

[0100] secretNamespace: kube-system

[0101] volumeoptions: user.heketi.arbiter true

[0102] provisioner: kubernetes.io/glusterfs

[0103] reclaimPolicy: Delete

[0104] volumeBindingMode: Immediate

[0105] allowVolumeExpansion: true。

[0106] 随后,执行步骤130,根据第一架构设置应用运行时所需的网络资源;具体的:获取第一镜像文件,第一镜像文件适于由第一架构的处理器运行;如果安装网络组件的所需镜像未适配第一架构,则根据第一架构构建第二镜像文件;根据第一镜像文件和第二镜像文件安装网络组件;根据网络组件设置应用运行时所需的网络资源。

[0107] 根据本发明的一个实施例,第一镜像文件为适配了第一架构的第一处理器的镜像文件。获取第一镜像文件可通过执行如下代码实现:

[0108] docker pull<对应镜像>

[0109] harbor.loongnix.cn/mirrorloongsoncontainers/calico/cni@sha256:90d8d53e423f673388b4c3e0237f3d386e3410d145f21bd778bace7852b295be

[0110] harbor.loongnix.cn/mirrorloongsoncontainers/calico/node@sha256:488526a50fb27a901e69bd27da6314826688a72d675e39669d48ade2c59a82b

[0111] harbor.loongnix.cn/mirrorloongsoncontainers/calico/kube-controllers@sha256:4785f1f6555441191e23079c197b83a986d4937538f3f793b05c0334fc0c5ee9。

[0112] 根据第一架构构建第二镜像文件时,先获取网络组件所需的一个或多个库文件;再根据一个或多个库文件,以及第一架构构建第二镜像文件。

[0113] 根据本发明的一个实施例,网络组件可实现为calico组件,本发明对网络组件的具体实现方式不做限制。Calico组件是Kubernetes中一种网络实现选择,不仅提供主机和pod之间的网络连接,还涉及网络安全和管理。其中,Calico CNI插件在CNI框架内封装了Calico的功能。

[0114] 根据本发明的一个实施例,先获取网络组件所需的一个或多个库文件,可通过执行如下代码实现:

[0115] git clone https://github.com/projectcalico/calico.git -b release-v3.18 ./calicoctl

[0116] cd calicoctl

[0117] old=`pwd`

[0118] go mod download #。

[0119] 根据本发明的一个实施例,判断一个或多个库文件是否适配了第一架构。若为适配第一架构,则需要重新构建该库文件。具体的:网络组件所需的库文件可实现为gopsutil库,gopsutil库未适配第一架构的第一处理器,因此,将所需要的gopsutil库的路径进行修改,使用本地可用的gopsutil库,具体可通过执行如下代码实现:

[0120] cd /root/go/pkg/mod/github.com/shirou/gopsutil@v0.0.0-20190323131628-2cbc9195c892/#。

[0121] 根据一个或多个库文件,以及第一架构构建第二镜像文件时,可具体通过执行如下代码实现:

[0122] DIRS="cpu disk docker host load mem net process"

[0123] GOOS="linux"

[0124] GOARCH="loong64"

[0125] for DIR in $DIRS

[0126] do

[0127] if [-e ${DIR}/types_${GOOS}.go];then

[0128] echo "// +build $GOOS">${DIR}/${DIR}_${GOOS}_${GOARCH}.go

[0129] echo "// +build $GOARCH">>${DIR}/${DIR}_${GOOS}_${GOARCH}.go

[0130] go tool cgo -godefs ${DIR}/types_${GOOS}.go

[0131] >>${DIR}/${DIR}_${GOOS}_${GOARCH}.go

[0132] fi

[0133] done**

[0134] cd $old CGO_ENABLED=0

[0135] go build-v-o bin/calicoctl -ldflags "-X

[0136] github.com/projectcalico/calicoctl/calicoctl/commands.VERSION=71e1f9726f82-X github.com/projectcalico/calicoctl/calicoctl/commands.GIT_REVISION=71e1f97-s-w"./calicoctl/calicoctl.go"

[0137] cat>Dockerfile<<'EOF'

[0138] FROM loong/alpine:v3.11.11

[0139] ADD calicoctl /

[0140] RUN chmod +x /calicoctl

[0141] ENV CALICO_CTL_CONTAINER=TRUE

[0142] ENTRYPOINT ["./calicoctl"]。

[0143] 根据本发明的一个实施例,接着根据第一镜像文件和第二镜像文件安装网络组件calico,并根据网络组件calico设置应用运行所需要的网络资源。

[0144] 随后,执行步骤140,根据依赖文件编译应用,并根据存储资源和网络资源运行应用。

[0145] 根据本发明的一个实施例,构建的依赖文件使应用能够适于第一架构的第一处理器。应用使用设置的存储资源和网络资源能够正常运行。

[0146] 根据本发明的一个实施例,计算设备与一个或多个其他计算设备通信连接,本发明方法还包括:在其他计算设备中构建一个或多个备份存储资源;当计算设备所提供的存

储资源不可用时,应用根据备份存储资源运行。

[0147] 在其他计算设备中构建一个或多个备份存储资源时,在其他计算设备中设置存储节点,并根据存储节点设置备份存储资源。

[0148] 根据本发明的一个实施例,设置存储节点时,可通过分布式存储系统进行设置。本发明对分布式存储系统的具体形式不作限制,如Minio存储系统。分布式存储系统通过将多个硬件存储设备组成一个对象存储服务。由于硬件存储设备分布在不同的节点上,避免了单点故障引起了存储失效。

[0149] 在计算设备或其他计算设备中构建存储节点时,根据计算设备或其他计算设备的架构类型获取安装包。

[0150] 随后配置安装环境,下载依赖的库文件,并进行编译,具体可通过执行如下代码实现:

[0151] go env -w GOPROXY='https://goproxy.cn'

[0152] go mod download

[0153] make build。

[0154] 在构建存储节点时,判断安装包是否支持当前计算设备的架构类型;若不支持出现报错,则检查安装包,确定依赖支持情况,再进行安装。具体的,可通过检查Makefile文件,调用checks函数检查依赖支持情况,再执行go build命令进行安装。

[0155] 根据本发明的一个实施例,报错的一个示例如下:

[0156] makefile

[0157] Checking dependencies

[0158] Arch 'loongarch64' is not supported. Supported Arch: [x86_64, amd64, ppc64le, aarch64, arm*, s390x]

[0159] make: *** [Makefile:17:checks] 错误 1。

[0160] 根据本发明的一个实施例,确定执行依赖可具体通过执行如下代码实现:

[0161] checks:

[0162] @echo "Checking dependencies"

[0163] @(env bash $(PWD)/buildscripts/checkdeps.sh)

[0164] build: checks

[0165] @echo "Building mc binary to './mc'"

[0166] @GO111MODULE=on CGO_ENABLED=0 go build -trimpath -tags kqueue -ldflags$(BUILD_LDFLAGS) -o $(PWD)/mc。

[0167] 根据本发明的一个实施例,依赖支持存在问题时,可跳过执行make build命令,直接通过go build命令构建可执行文件,具体可通过执行如下代码实现:

[0168] LDFLAGS := $(shell go run buildscripts/gen-ldflags.go)

[0169] BUILD_LDFLAGS := '$(LDFLAGS)'

[0170] PWD := $(shell pwd)。

[0171] 根据本发明的一个实施例,可通过执行如下代码,执行go build命令构建可执行文件:

[0172] GO111MODULE=on CGO_ENABLED=0 go build -trimpath -tags kqueue --

ldflags $(gorun buildscripts/gen-ldflags.go) -o $(pwd)/mc。

[0173] 根据本发明的一个实施例,本发明还对应用是否能够正常运行进行测试。具体的:设置默认后端镜像和控制器镜像;根据默认后端镜像和控制器镜像设置负载均衡服务;根据负载均衡服务确定应用是否运行正常。

[0174] 根据本发明的一个实施例,默认后端镜像可具体实现为defaultbackend镜像,设置默认后端镜像时,获取未适配的默认后端镜像文件,未适配的默认后端镜像文件包括makefile文件和dockerfile文件;具体可执行如下代码:

[0175] git clone http://*/caas-go/default-http-backend.git -b main&&cd default-http-backend。

[0176] 随后编译未适配的默认后端镜像文件;具体可执行如下代码:

[0177] 《makefile》

[0178] GOARCH=$(shell go env GOARCH)

[0179] .PHONY: clean server image

[0180] clean:

[0181] rm -f build/server

[0182] server: main.go

[0183] CGO_ENABLED=0 GOOS=linux GOARCH=${GOARCH} go build -o build/server main.go

[0184] image: clean server

[0185] docker build -t defaultbackend:1.0 ./build

[0186] 《dockerfile》

[0187] FROM alpine:v3.11.11

[0188] COPY server /bin/server

[0189] ENTRYPOINT ["/bin/server"]。

[0190] 最后修改未适配的默认后端镜像文件的文件标识以便适配第一架构;具体的可执行如下代码:

[0191] docker tag defaultbackend:1.0 k8s-deploy-loong64/defaultbackend:1.0。

[0192] 根据本发明的一个实施例,控制器镜像可具体实现为nginx-ingress-controller镜像。设置控制器镜像时,预先获取未适配的控制器镜像文件,对控制器镜像文件进行适配;具体的:先确定未适配的控制器镜像文件中的缺失函数。通过获取未适配的控制器镜像文件,并对未适配的控制器镜像文件进行测试启动,根据报错信息确定缺失函数。根据本发明的一个实施例,缺失函数包括nginx.tmpl函数。

[0193] 随后,在未适配的控制器镜像文件中构建缺失函数,对控制器镜像文件进行适配;具体的可通过执行如下代码实现:

[0194] docker cp xxxxx:/etc/nginx/template/nginx.tmpl /root/。

[0195] 根据本发明的一个实施例,根据默认后端镜像和控制器镜像设置的负载均衡服务,具体可实现为一种负载均衡器(Nginx ingress controller)。负载均衡服务用于为应用提供服务,是应用服务的外部访问进行管理的工具。

[0196] 根据负载均衡服务确定应用是否运行正常时,可通过负载均衡服务确定应用的运

行信息,如流量流转过程和调用各服务的过程等;通过应用的运行信息即可确定应用是否运行正常。

[0197] 本发明公开了一种应用运行方法,适于在计算设备中执行,计算设备中包括第一处理器,第一处理器的架构类型为第一架构,方法包括步骤:根据第一架构构建应用编译时所需的依赖文件,应用与第二处理器相适配,第二处理器的架构类型为第二架构;根据第一架构设置应用运行时所需的存储资源;根据第一架构设置应用运行时所需的网络资源;根据依赖文件编译应用,并根据存储资源和网络资源运行应用。本发明中通过对应用编译所需文件进行构建,实现在未适配相应处理器架构的计算设备中能够启动应用,并且配置所需的存储资源和网络资源,实现应用在未适配相应处理器架构的计算设备中运行。

[0198] 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。

[0199] 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。

[0200] 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间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。

[0201] 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间组合成一个模块或单元或组间,以及此外可以把它们分成多个子模块或子单元或子组间。除了这样的特征和/或过程或者单元中的至少一些是相互排斥之外,可以采用任何组合对本说明书中公开的所有特征以及如此公开的任何方法或者设备的所有过程或单元进行组合。除非另外明确陈述,本说明书中公开的每个特征可以由提供相同、等同或相似目的的替代特征来代替。

[0202] 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。

[0203] 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。

[0204] 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,例如软盘、CD-ROM、硬盘驱动器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。

[0205] 在程序代码在可编程计算机上执行的情况下，计算设备一般包括处理器、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，执行本发明的应用运行方法。

[0206] 以示例而非限制的方式，计算机可读介质包括计算机存储介质和通信介质。计算机可读介质包括计算机存储介质和通信介质。计算机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计算机可读介质的范围之内。

[0207] 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。

[0208] 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。因此，对于本技术领域的普通技术人员来说许多修改和变更都是显而易见的。对于本发明的范围，对本发明所做的公开是说明性的，而非限制性的。

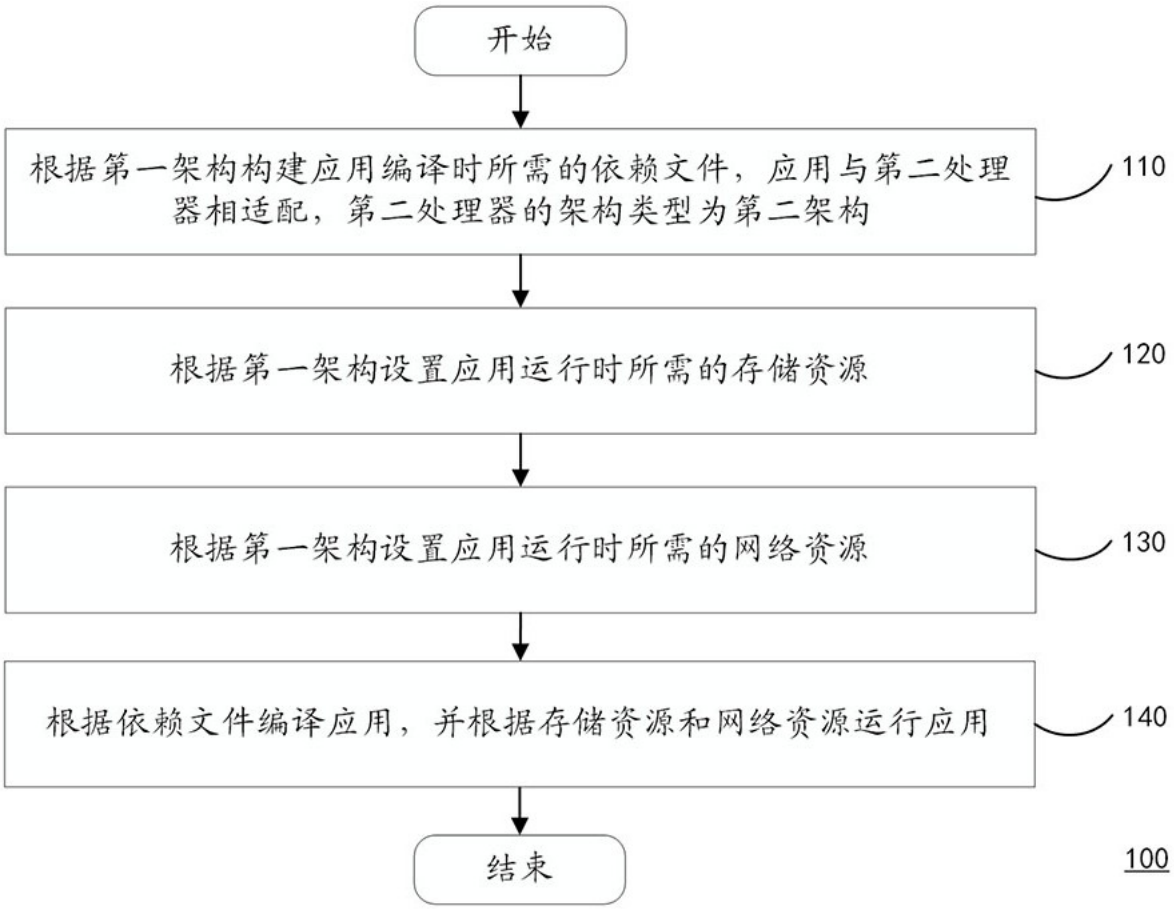

开始
根据第一架构构建应用编译时所需的依赖文件，应用与第二处理器相适配，第二处理器的架构类型为第二架构
110
根据第一架构设置应用运行时所需的存储资源
120
根据第一架构设置应用运行时所需的网络资源
130
根据依赖文件编译应用，并根据存储资源和网络资源运行应用
140
结束
100

图 1

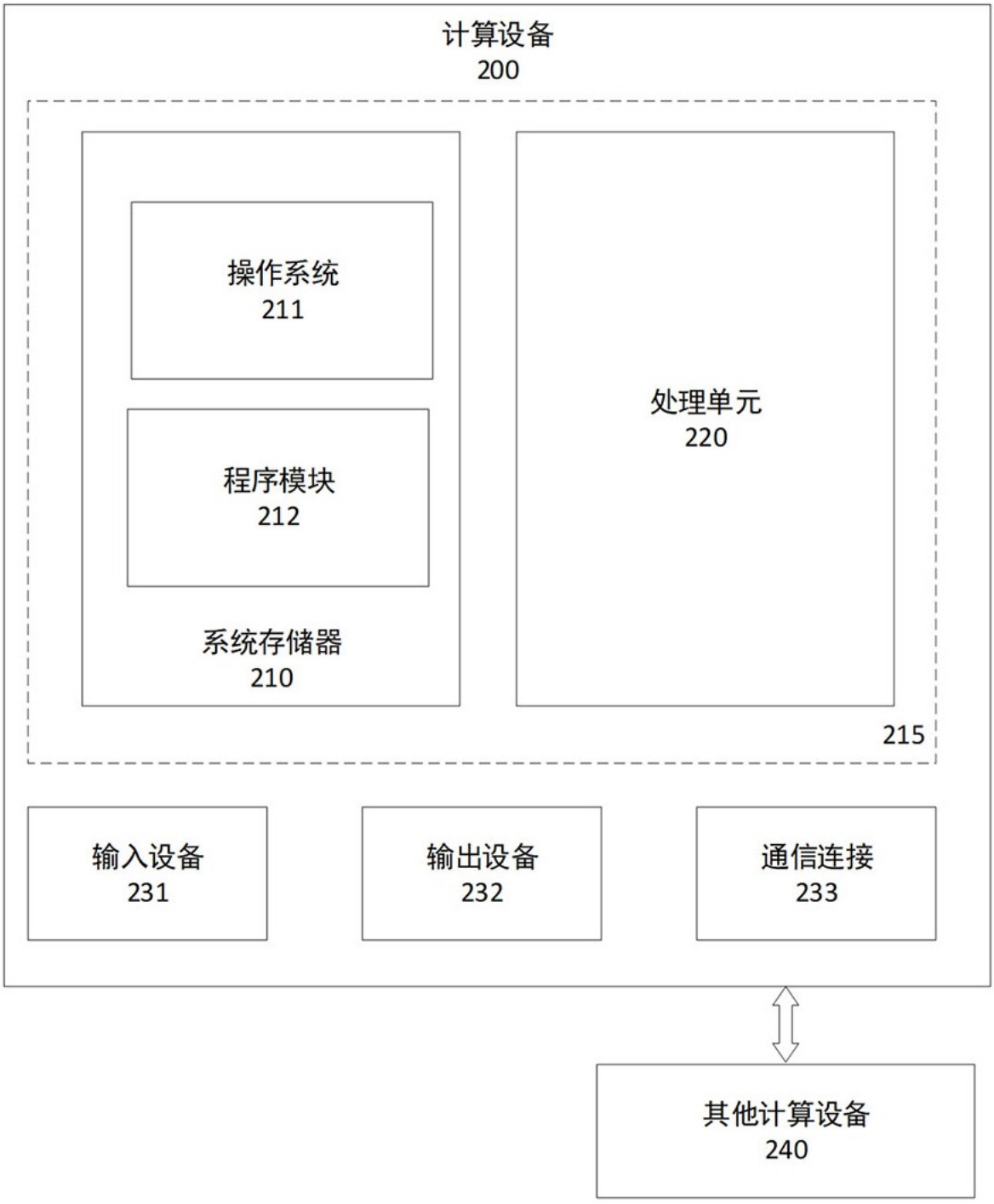

计算设备
200
操作系统
211
程序模块
212
系统存储器
210
处理单元
220
215
输入设备
231
输出设备
232
通信连接
233
其他计算设备
240

图 2