(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 114489947 B
(45)授权公告日 2025.05.30

(21)申请号 202210096155.4
(22)申请日 2022.01.26
(65)同一申请的已公布的文献号
申请公布号 CN 114489947 A
(43)申请公布日 2022.05.13
(73)专利权人 阿里巴巴(中国)有限公司
地址 310052 浙江省杭州市滨江区长河街道网商路699号4号楼5楼508室
(72)发明人 王骜 徐亦达 常率 王宏琦 杨皓然
(74)专利代理机构 北京太合九思知识产权代理有限公司 11610
专利代理师 邓春燕
(51)Int.Cl.
G06F 9/455(2006.01)

(56)对比文件
WO 2020226659 A1,2020.11.12

审查员 高秀攀

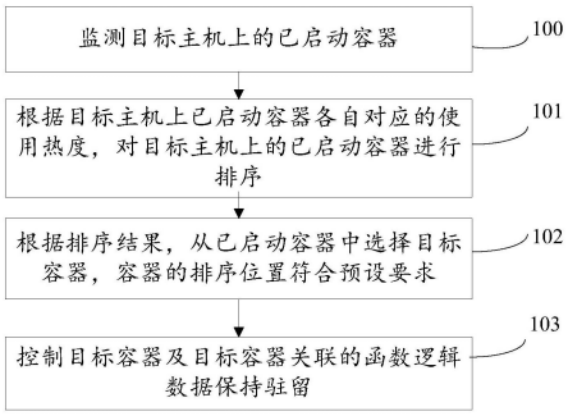
权利要求书2页 说明书13页 附图4页

(54)发明名称

容器的生命周期管理、函数计算方法、设备及存储介质

(57)摘要

本申请实施例提供一种容器的生命周期管理、函数计算方法、设备及存储介质。可监测目标主机上的已启动容器;实时地计算各个已启动容器的使用热度;并根据使用热度,实时地维护目标主机上的已启动容器之间的排序;基于已启动容器之间的排序,控制排序位置符合预设要求的目标容器保持驻留,直至目标容器因排序位置变动而不再符合预设要求。这样,容器的生命周期不再死板地遵循固定的阈值时长,而是可按照缓存的思想,控制更有可能被用到的容器及其关联的函数逻辑数据驻留更长时间,从而有效提高使用热启动的方式来执行函数的概率,而减少甚至避免冷启动;热启动的延时要远远低于冷启动的延时,因此,整体上可有效提高函数的执行效率。



CN 114489947 B

1.一种容器的生命周期管理方法,包括:

监测目标主机上的已启动容器;

根据所述目标主机上已启动容器各自对应的使用热度,对所述目标主机上的已启动容器进行排序;

根据排序结果,从已启动容器中选择目标容器,所述目标容器的排序位置符合预设要求;

控制所述目标容器及所述目标容器关联的函数逻辑数据保持驻留,直至所述目标容器因排序位置变动而不再符合所述预设要求。

2.根据权利要求1所述的方法,还包括:

分别确定各个已启动容器的启动延迟、所占用内存的规格和/或历史驻留时间;

其中,对所述目标主机上的已启动容器进行排序的依据除所述使用热度外,还包括所述启动延迟、所述所占用内存的规格和/或所述历史驻留时间。

3.根据权利要求2所述的方法,所述对所述目标主机上的已启动容器进行排序,包括:

计算所述目标主机上的已启动容器各自对应的优先级;

按照所述优先级对所述目标主机上的已启动容器进行排序;

其中,所述优先级与所述使用热度成正比;所述优先级与所述启动延迟成正比;所述优先级与所述所占内存的规格成反比;所述优先级与所述历史驻留时间成正比。

4.根据权利要求1所述的方法,所述根据排序结果,从已启动容器中选择目标容器,包括:

分别为各个已启动容器构建容器画像;

为所述目标主机维护容器队列;

按照所述排序结果,确定各个已启动容器的容器画像在所述容器队列中的排序位置;

从所述容器队列中,选择排序位置符合所述预设要求的容器画像对应的已启动容器,作为所述目标容器。

5.根据权利要求4所述的方法,所述选择排序位置符合所述预设要求的容器画像对应的已启动容器,作为所述目标容器,包括:

将位于所述容器队列之内的容器画像对应的已启动容器,确定为所述目标容器。

6.根据权利要求1所述的方法,还包括:

接收针对目标函数的函数触发指令;

若所述目标函数被调度至所述目标主机上目标容器中的指定容器,则将所述指定容器的排序位置前移;

基于留存的所述目标函数对应的函数逻辑数据,利用所述指定容器执行所述目标函数。

7.根据权利要求1所述的方法,包括:

在目标用户发起的针对目标函数的创建/更新指令的情况下,若所述创建/更新指令被调度至所述目标主机上且所述目标主机上不存在可负载所述目标函数的容器,则在所述目标用户符合指定条件时,控制所述目标主机预先获取所述目标函数对应的函数逻辑数据并留存在所述目标主机上。

8.根据权利要求7所述的方法,还包括:

在接收到针对所述目标函数的函数触发指令时,控制所述目标主机为所述目标函数创建容器;

基于所述目标主机上预先获取的所述目标函数对应的函数逻辑数据,利用为所述目标函数创建的容器执行所述目标函数。

9.根据权利要求7所述的方法,所述目标主机与集群中的其它主机处于对等网络中,所述控制所述目标主机预先获取所述目标函数对应的函数逻辑数据,包括:

控制所述目标主机优先从所述对等网络中的其它主机上获取所述目标函数对应的函数逻辑数据。

10.根据权利要求1所述的方法,还包括:

控制已启动容器中排序位置不符合所述预设要求的容器释放;

删除已释放容器关联的函数逻辑数据。

11.一种函数计算方法,包括:

接收针对目标函数的函数触发请求;

若所述目标函数被调度至目标主机上已启动的指定容器,则将所述指定容器在所述目标主机上已启动的所有容器中所处的排序位置进行前移,以延长所述指定容器的驻留时间;

基于所述目标函数对应的函数逻辑数据,利用所述指定容器执行所述目标函数,所述目标函数对应的函数逻辑数据预先留存于所述目标主机上。

12.根据权利要求11所述的方法,还包括:

接收针对所述目标函数的创建/更新指令;

若所述创建/更新指令被调度至所述目标主机上且所述目标主机上不存在可负载所述目标函数的容器,则在所述创建/更新指令的发起方符合指定条件时,控制所述目标主机预先获取所述目标函数对应的函数逻辑数据并留存在所述目标主机上。

13.一种路由设备,包括存储器、处理器和通信组件;

所述存储器用于存储一条或多条计算机指令;

所述处理器与所述存储器和所述通信组件耦合,用于执行所述一条或多条计算机指令,以用于:

通过所述通信组件监测目标主机上的已启动容器;

根据所述目标主机上已启动容器各自对应的使用热度,对所述目标主机上的已启动容器进行排序;

根据排序结果,从已启动容器中选择目标容器,所述目标容器的排序位置符合预设要求;

控制所述目标容器及所述目标容器关联的函数逻辑数据保持驻留,直至所述目标容器因排序位置变动而不再符合所述预设要求。

14.一种存储计算机指令的计算机可读存储介质,当所述计算机指令被一个或多个处理器执行时,致使所述一个或多个处理器执行权利要求1-10任一项所述的容器的生命周期管理方法或权利要求11或12所述的函数计算方法。

容器的生命周期管理、函数计算方法、设备及存储介质

技术领域

[0001] 本申请涉及云计算技术领域,尤其涉及容器的生命周期管理、函数计算方法、设备及存储介质。

背景技术

[0002] FaaS(Function-as-a-Service,功能即服务或者函数计算服务),为开发人员提供了一种方便的方式来执行他们编写的函数逻辑。这样,开发人员只需要专注于他们的逻辑实现,而如集群管理、资源配置和虚拟化技术等其他工作则都由云提供商负责。

[0003] 尽管FaaS可以为用户提供一种方便的方式,但是由于函数逻辑的执行需依赖容器支持,而目前的容器生命周期管理方案比较死板,因此,函数逻辑的执行经常需要等待容器启动。这样,容器的启动延时会损害函数逻辑的执行效率,尤其是在高并发的情况下,严重影响用户的体验。

发明内容

[0004] 本申请的多个方面提供一种容器的生命周期管理、函数计算方法、设备及存储介质,用以提高依赖容器的函数的执行效率。

[0005] 本申请实施例提供一种容器的生命周期管理方法,包括:

[0006] 监测目标主机上的已启动容器;

[0007] 根据所述目标主机上已启动容器各自对应的使用热度,对所述目标主机上的已启动容器进行排序;

[0008] 根据排序结果,从已启动容器中选择目标容器,所述目标容器的排序位置符合预设要求;

[0009] 控制所述目标容器及所述目标容器关联的函数逻辑数据保持驻留。

[0010] 本申请实施例还提供一种函数计算方法,包括:

[0011] 接收针对目标函数的函数触发请求;

[0012] 若所述目标函数被调度至目标主机上已启动的指定容器,则将所述指定容器在所述目标主机上已启动的所有容器中所处的排序位置进行前移,以延长所述指定容器的驻留时间;

[0013] 基于所述目标函数对应的函数逻辑数据,利用所述指定容器执行所述目标函数,所述目标函数对应的函数逻辑数据预先留存于所述目标主机上。本申请实施例还提供一种路由设备,包括存储器、处理器和通信组件;

[0014] 所述存储器用于存储一条或多条计算机指令;

[0015] 所述处理器与所述存储器和所述通信组件耦合,用于执行所述一条或多条计算机指令,以用于:

[0016] 通过所述通信组件监测目标主机上的已启动容器;

[0017] 根据所述目标主机上已启动容器各自对应的使用热度,对所述目标主机上的已启

动容器进行排序;

[0018] 根据排序结果,从已启动容器中选择目标容器,所述目标容器的排序位置符合预设要求;

[0019] 控制所述目标容器及所述目标容器关联的函数逻辑数据保持驻留。

[0020] 本申请实施例还提供一种存储计算机指令的计算机可读存储介质,当所述计算机指令被一个或多个处理器执行时,致使所述一个或多个处理器执行前述的容器的生命周期管理方法。

[0021] 在本申请实施例中,可监测目标主机上的已启动容器;实时地计算各个已启动容器的使用热度;并根据使用热度,实时地维护目标主机上的已启动容器之间的排序;基于已启动容器之间的排序,控制排序位置符合预设要求的目标容器保持驻留,直至目标容器因排序位置变动而不再符合预设要求。这样,容器的生命周期不再死板地遵循固定的阈值时长,而是可按照缓存的思想,控制更有可能被用到的容器及其关联的函数逻辑数据驻留更长时间,从而有效提高使用热启动的方式来执行函数的概率,而减少甚至避免冷启动;热启动的延时要远远低于冷启动的延时,因此,整体上可有效提高函数的执行效率。

附图说明

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

[0023] 图1为本申请一示例性实施例提供的一种容器的生命周期管理方法的流程示意图;

[0024] 图2为本申请一示例性实施例提供的一种容器的生命周期管理方案的逻辑示意图;

[0025] 图3为本申请实施例提供的一种示例性的容器队列的示意图;

[0026] 图4为本申请一示例性实施例提供的一种预取方案的逻辑示意图;

[0027] 图5为本申请另一示例性实施例提供的一种函数计算方法的流程示意图;

[0028] 图6为本申请又一示例性实施例提供的一种路由设备的结构示意图。

具体实施方式

[0029] 为使本申请的目的、技术方案和优点更加清楚,下面将结合本申请具体实施例及相应的附图对本申请技术方案进行清楚、完整地描述。显然,所描述的实施例仅是本申请一部分实施例,而不是全部的实施例。基于本申请中的实施例,本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例,都属于本申请保护的范围。

[0030] 目前,容器生命周期管理方案比较死板,因此,函数逻辑的执行经常需要等待容器启动,存在大量的冷启动延时,严重损害函数逻辑的执行效率。为改善这些技术问题,本申请的一些实施例中:可监测目标主机上的已启动容器;实时地计算各个已启动容器的使用热度;并根据使用热度,实时地维护目标主机上的已启动容器之间的排序;基于已启动容器之间的排序,控制排序位置符合预设要求的目标容器保持驻留,直至目标容器因排序位置变动而不再符合预设要求。这样,容器的生命周期不再死板地遵循固定的阈值时长,而是可按照缓存的思想,控制更有可能被用到的容器及其关联的函数逻辑数据驻留更长时间,从

而有效提高使用热启动的方式来执行函数的概率,而减少甚至避免冷启动;热启动的延时要远远低于冷启动的延时,因此,整体上可有效提高函数的执行效率。

[0031] 以下结合附图,详细说明本申请各实施例提供的技术方案。

[0032] 图1为本申请一示例性实施例提供的一种容器的生命周期管理方法的流程示意图。本实施例提供的容器的生命周期的管理方法可以由一管控装置来执行,该管控装置可以实现为软件或实现为软件和硬件的组合,该管控装置可集成设置在路由设备中。如图1所示,该容器的生命周期管理方法,可包括:

[0033] 步骤100、监测目标主机上的已启动容器;

[0034] 步骤101、根据目标主机上已启动容器各自对应的使用热度,对目标主机上的已启动容器进行排序;

[0035] 步骤102、根据排序结果,从已启动容器中选择目标容器,目标容器的排序位置符合预设要求;

[0036] 步骤103、控制目标容器及目标容器关联的函数逻辑数据保持驻留。

[0037] 本实施例提供的容器的生命周期管理方法可应用于FaaS场景,对FaaS场景中的容器进行更加合理地管控,以更高效地进行函数计算。FaaS是Functions as a Service的缩写,可以广义的理解为功能服务化,也可以解释为函数计算服务化。使用FaaS只需要关注函数代码逻辑,无需关注服务器资源,可以说FaaS提供了一个更加细分和抽象的服务化能力。本实施例提供的方法的执行主体可以是FaaS架构中的路由设备,当然,本实施例中的执行主体并不限于此,例如,执行主体还可以是FaaS架构中的其它设备、管控节点或新增节点等。

[0038] 图2为本申请一示例性实施例提供的一种容器的生命周期管理方案的逻辑示意图。参考图2可知,路由设备可与多个主机host通信,并管控多个主机上的容器。应当理解的是,本实施例提供的容器的生命周期管理方案主要涉及图2中路由设备与多个主机,因此,在图2中并未示出FaaS架构中其它组件、节点等成员,本实施例对FaaS架构中其它成员的连接方式不作限定,可参考当前或将来出现的各种连接方式。另外,正如前文提及的,图2中的路由设备也可替换为其它控制设备,可用于对其关联的各个主机上的容器进行管控的设备均可。

[0039] 为便于描述,本实施例中,将以路由设备作为示例性的执行主体,以其所通信的多个主机中的目标主机为例,进行技术方案的说明,应当理解的是,目标主机可以是路由设备所通信的多个主机中的任意一个。另外,本实施例中,主机的实现形式可以是、云服务器、虚拟机等等,本实施例对此不做限定。用户编写的函数逻辑数据可由容器来执行,而容器将与其执行过的函数逻辑数据关联,本实施例中,容器与其执行过的函数逻辑数据可绑定在一起,同步驻留,同步释放。其中,函数逻辑数据可以是用户提供的,例如,可以是函数代码。举例来说,企业用户可指示其开发人员编写用于查询天气的函数代码,并将函数代码存放在云盘等外部存储组件中,容器可运行函数代码以执行“查询天气”函数。

[0040] 基于此,参考图1和图2,在步骤100中,可监测目标主机上的已启动容器。其中,已启动容器是指已经映射至实际物理资源的容器。但是应当理解的是,已启动容器上可能已经负载有函数function,也可能并未负载任何函数function,也即是,已启动容器可能处于工作中状态,也可能处于空闲状态,本实施例对已启动容器的状态无限制。目标主机上的已

启动容器可能是一个或多个。

[0041] 在步骤101中,可根据目标主机上已启动容器各自对应的使用热度,对目标主机上的已启动容器进行排序。本实施例中,可分别确定各个已启动容器的使用热度:可根据各个已启动容器的被请求频率来计算使用热度,使用热度与被请求频率成正比。为此,本实施例中,可记录各个已启动容器的历史被请求次数,以统计其被请求频率。其中,随着时间的变化,目标主机上的已启动容器会发生变化,已启动容器的使用热度也会发生变化,本实施例中,可周期性地或者基于后文提及的函数触发请求作为触发,来确定目标主机上的已启动容器的使用热度,以保持使用热度的实时性。基于此,可根据使用热度,来对目标主机上的已启动容器进行排序。值得说明的是,本实施例中,使用热度仅是对目标主机上的已启动容器进行排序的依据之一。在不考虑其它排序依据的情况下,本实施例中认为使用热度越高的已启动容器在后续更有可能被再次使用,因此,通过对目标主机上的已启动容器进行排序,可反应出哪些已启动容器在后续更有可能被再次使用。

[0042] 在此基础上,在步骤102中,可根据排序结果,从已启动容器中选择目标容器,目标容器的排序位置符合预设要求。其中,随着时间的变化,目标主机上的已启动容器会发生变化,已启动容器之间的排序也会发生变化,本实施例中,可随时间不断更新已启动容器之间的排序。一种示例性的排序方案可以是:后续被再次使用的可能性越高的排序位置越靠前,当然,这仅是示例性的,本实施例并不限于此。

[0043] 基于已启动容器之间的排序结果,在步骤102中,可通过预设要求来确定出哪些已启动容器需要保持驻留,哪些已启动容器需要释放。并在步骤103中,控制目标容器及目标容器关联的函数逻辑数据保持驻留,直至目标容器因排序位置变动而不再符合预设要求。其中,预设要求可根据实际需要进行设定,设定的预设要求能够使更有可能被再次使用的容器驻留更长的时间即可。随着已启动容器的迭代更新以及已启动容器之间排序的迭代更新,后续更可能被再次使用的已启动容器将不断地被驻留下来,也即是,这类已启动容器的驻留时间将更长,而这类已启动容器所关联的函数function通常也是那些高请求频率的函数,这就形成了良性循环,接收到函数触发请求后,将有更大概率可使用目标主机上驻留的目标容器来执行对应的函数,而不再需要启动新的容器,而且驻留的目标容器所关联的函数逻辑数据也被同步留存了,因此,也不再需要去远程下载函数逻辑数据,这使得函数的执行几乎不存在延时,效率非常高。

[0044] 本实施例中,可将容器的启动类型划分为至少三种:

[0045] 一种是热启动,若主机上存在可负载目标函数的容器且相关的函数逻辑数据已留存,则只需将函数触发请求调配至相应的容器上即可,这种方式称为热启动,热启动方式的延时非常小。

[0046] 一种是冷启动,若主机上不存在可负载目标函数的容器,也未留存有相关的函数逻辑数据,则需要下载相关的函数逻辑数据并启动新的容器,这种方式称为冷启动,冷启动方式的延时很大,可达到秒级甚至分钟级。一个完整的冷启动过程可包括但不限于函数逻辑数据准备时间、运行环境初始化时间和容器启动时间等。

[0047] 还有一种是半冷启动,若主机上不存在可负载目标函数的容器,但是留存有相关的函数逻辑数据,则可只需为目标函数启动新的容器即可,与冷启动相比可节省下载相关的函数逻辑的时间,这种方式称为半冷启动,半冷启动方式的延时比冷启动方式的延时要

小,但还是大于热启动方式的延时。

[0048] 本实施例中,还可控制已启动容器中排序位置不符合预设要求的容器释放;还可删除已释放容器关联的函数逻辑数据,以节省目标主机上的计算资源和存储资源。

[0049] 这样,本实施例中,可监测目标主机上的已启动容器;实时地计算各个已启动容器的使用热度;并根据使用热度,实时地维护目标主机上的已启动容器之间的排序;基于已启动容器之间的排序,控制排序位置符合预设要求的目标容器保持驻留,直至目标容器因排序位置变动而不再符合预设要求。这样,容器的生命周期不再死板地遵循固定的阈值时长,而是可按照缓存的思想,控制更有可能被用到的容器及其关联的函数逻辑数据驻留更长时间,从而有效提高使用热启动的方式来执行函数的概率,而减少甚至避免冷启动;热启动的延时要远远低于冷启动的延时,因此,整体上可有效提高函数的执行效率。

[0050] 在上述或下述实施例中,对目标主机上的已启动容器进行排序的依据除使用热度外,还可包括但不限于启动延迟、所占用内存的规格或历史驻留时间等,这些依据共同影响已启动容器之间的排序。为此,本实施例中,还可分别为各个已启动容器确定启动延迟、所占用内存的规格和/或历史驻留时间。其中,启动延迟可以是指已启动容器启动所需耗费的时长。所占用内存的规格可以是指为已启动容器配置的内存空间大小,不同的已启动容器所配置的内存空间大小可能不同,这可由用户来决定,比如用户可在函数的创建指令中指定所需的内存空间大小,为该函数创建容器时将按照指定的内存空间大小进行配置。历史驻留时间可以是指容器已驻留的时长。

[0051] 基于此,本实施例中,可基于这些排序依据,对目标主机上的已启动容器进行排序。

[0052] 本实施例中可采用多种实现方式来执行排序操作。

[0053] 在一种实现方式中,可计算目标主机上的已启动容器各自对应的优先级;按照优先级对目标主机上的已启动容器进行排序。其中,优先级与使用热度成正比;优先级与启动延迟成正比;优先级与所占内存的规格成反比;优先级与历史驻留时间成正比。

[0054] 在考虑上述多种排序依据的情况下,本实施例中,已启动容器的优先级的计算过程可表征为:

[0055] $$Priority = Freq * \frac{Cost}{Size} * TTL$$

[0056] 其中,Priority表示优先级,Freq表示使用热度Frequency,Cost表示启动延迟StartUpLatency,Size表示所占内存的规格MemorySize,TTL表示历史驻留时间Time-To-Live。

[0057] 这样,目标主机上的已启动容器将获得动态变化的优先级,而基于动态变化的优先级,已启动容器之间的排序也将动态变化。一种示例性的排序方式可以是,优先级越高的已启动容器的排序位置越靠前。

[0058] 应当理解的是,在上述的实现方式中,采用优先级来表征已启动容器在后续被再次使用的可能性。但本实施例并不限于,还可使用其它实现方式来执行排序操作,例如,根据前述的多种排序依据使用机器学习模型等预测已启动容器被再次使用的概率,按照该概率进行排序,等。

[0059] 据此,本实施例中,可按照各种排序依据,动态维护目标主机上已启动容器之间的

排序,从而可准确感知已启动容器在后续被再次使用的可能性,进而为容器的生命周期管理提供可靠的依据。

[0060] 在上述或下述实施例中,可采用多种实现方式来确定出已启动容器中可被保持驻留的目标容器。

[0061] 在一种实现方式中,可分别为各个已启动容器构建容器画像;为目标主机维护容器队列;按照对目标主机上的已启动容器进行排序的排序结果,确定各个已启动容器的容器画像在容器队列中的排序位置;从容器队列中,选择排序位置符合预设要求的容器画像对应的已启动容器,作为目标容器。

[0062] 其中,容器画像可用于抽象表征一个容器。容器画像中可包含但不限于容器的ID、所占用内存的规格、所执行函数逻辑数据的文件大小、已执行函数的执行时间和创建时间等元数据信息。

[0063] 图3为本申请实施例提供的一种示例性的容器队列的示意图。参考图3,路由设备可关联多台主机EE-1～EE-5,并分别为每条主机维护一条容器队列。本实施例对容器队列的规格不作限定,而且,不同主机可对应不同的容器队列规格。容器队列中的容器画像的排序位置可动态变化。

[0064] 承接上文实施例中提及的通过优先级的方式来表征容器后续被再次使用的可能性,结合图3,可按照优先级由高到低的顺序,将目标主机上已启动容器的容器画像依次放入目标主机对应的容器队列中。

[0065] 在该实现方式中,可基于容器队列的规格来设定前述的与排序位置相关的预设要求,一种示例性的预设要求可以是容器画像位于容器队列之内。基于此,在该实现是中,可将位于容器队列之内的容器画像对应的已启动容器,确定为目标容器;控制目标容器保持驻留,直至目标容器因排序位置变动而被挤出容器队列之外。参考图3,主机EE-1的容器队列已满,而有一个已启动容器因优先级过低被挤在容器队列之外,则该冗余的已启动容器将被释放,而位于容器队列之内的其它已启动容器则将被保留,而如果主机EE-1上启动了新的容器,而新的已启动容器的优先级比当前容器队列队尾的已启动容器的优先级要高,那当前容器队列队尾的已启动容器将被挤出容器队列之外,并被释放。

[0066] 当然,在该实现方式中,还可采用其它预设要求,例如,预设要求可以是容器画像位于容器队列的前半部分,或者容器画像位于容器队列前60%的位置等等,预设要求并不限于此。

[0067] 另外,本实施例中,还可采用其它实现方式来确定出已启动容器中可被保持驻留的目标容器,而并不限于上述的容器队列的方式。例如,可采用表格或其它可动态记录的载体来维护已启动容器之间的排序,等。

[0068] 据此,本实施例中,可采用容器队列等方式来动态维护目标主机上已启动容器之间的排序,并可方便地按需设定预设要求,来决定哪些已启动容器可被保持驻留,哪些已启动容器可被释放,使得容器的生命周期管理更加合理化。

[0069] 在上述或下述实施例中,还可接收针对目标函数的函数触发指令;若目标函数被调度至目标主机上目标容器中的指定容器,则将指定容器的排序位置前移;基于留存的目标容器关联的函数逻辑数据,利用指定容器执行目标函数。其中,函数触发指令用于指示开始执行目标函数。

[0070] 本实施例中,可由路由设备来进行函数触发指令的调度。举例来说,路由设备可从容器的维度来进行函数触发指令的调度:路由设备可将指定容器的位置信息配置到函数触发指令中,从而将函数触发指令调度至指定容器,其中位置信息中可包括但不限于容器所处主机的地址等。当然,这是在存在可负载目标函数的容器的情况下的调度方案;而在不存在可负载目标函数的容器的情况下路由设备可将函数触发指令调度至指定主机上,并由指定主机来为目标函数创建新的容器。对目标函数的一种示例性调度原则可以是:优先调度至驻留的容器;优先调度至上一次负载目标函数的容器上。

[0071] 本实施例中,目标函数被调度至目标主机上目标容器中的指定容器,则可由指定容器来运行目标函数对应的函数逻辑数据,以执行目标函数。其中,按照上述提及的示例性调度原则,目标函数对应的函数逻辑数据大概率已经留存在目标主机上,因此,目标主机可基于留存的目标函数对应的函数逻辑数据,利用指定容器执行目标函数。

[0072] 而由于指定容器的被请求次数发生了新增,因此,指定容器的排序位置可前移,当然,若其它已启动容器的被请求次数也发生变化,则还是要统筹考虑目标主机上所有已启动容器,来确定它们之间的排序。

[0073] 本实施例中,用户除了可发起函数触发指令外,还可能发起创建/更新指令。其中,创建指令用于指示创建函数,更新指令则用于指示更新函数。目前,这两个指令均是为了函数执行做准备工作,只有用户随后发起函数触发指令时,才会真正去执行对应的函数。

[0074] 本实施例中,对创建/更新指令进行了创新性的使用,具体地:可在目标用户发起的针对目标函数的创建/更新指令的情况下,若创建/更新指令被调度至目标主机上且目标主机上不存在可负载目标函数的容器,则在目标用户符合指定条件时,控制目标主机预先获取目标函数对应的函数逻辑数据并留存在目标主机上。也即是,在目标用户发起针对目标函数的创新/更新指令且目标用户符合指定条件的情况下,可认为目标函数即将发生执行需求,并可通过预取技术,提前将目标函数对应的函数逻辑数据下载到目标主机上,以备用。

[0075] 这里,通过指定条件对预取工作进行了限制,例如,指定条件可以是用户后续发起函数触发指令的可能性符合指定标准。为此,在一种示例性方案中,可根据目标用户的历史行为记录,统计目标用户针对同一函数发起创建/更新指令与发起函数触发指令之间的时间差,以表征用户后续发起函数触发指令的可能性,这样,时间差小于指定标准的情况下,才确定目标用户符合指定条件,这种情况下,才执行预取目标函数对应的函数逻辑数据的操作。

[0076] 图4为本申请一示例性实施例提供的一种预取方案的逻辑示意图。参考图4,目标用户发起的针对目标函数的创建/更新指令可首先到达FaaS对应集群中的API服务器,API服务器可按常规方案对创建/更新指令进行响应,这部分是现有方案不再展开说明。本实施例中的创新使用在于,控制API服务器将创建/更新指令通知给路由设备,这样,路由设备可获知目标用户发起了针对目标函数的创建/更新指令,并可为目标函数选择调度至的主机,例如是目标主机,若目标主机上不存在可负载目标函数的容器且目标用户符合指定条件,则可控制目标主机为目标函数执行预取操作并留存目标函数对应的函数逻辑数据。当然,路由设备还可从其它渠道获知目标用户发起的针对目标函数的创建/更新指令,本实施例并不限于此。这样,在获取目标用户发起了针对目标函数的创建/更新指令且目标用户符合

指定条件的情况下,可预先准备好目标函数对应的函数逻辑数据。

[0077] 基于此,本实施例中,可在接收到针对目标函数的函数触发指令时,控制目标主机为目标函数创建容器;基于目标主机上预先获取的目标函数对应的函数逻辑数据,利用为目标函数创建的容器执行目标函数。也即是,在完成上述的预取工作后,后续一旦接收到针对目标函数的函数触发指令,则只需启动新的容器即可,而不再需要远程下载对应的函数逻辑数据,从而,可实现按照前述的半冷启动方式来执行目标函数,这相比于冷启动方式可节省大量的时延。

[0078] 参考图4,在一种示例性的预取方案中:目标主机与集群中的其它主机可处于对等网络中,基于此,可控制目标主机优先从对等网络中的其它主机上获取目标函数对应的函数逻辑数据。也即是,可通过P2P的方式,控制目标主机优先从对等网络中的其它主机中预取目标函数对应的函数逻辑数据,当然,在对等网络中不存在目标函数对应的函数逻辑数据,目标主机可从源头(云盘等外部存储组件)中预取目标函数对应的函数逻辑数据。

[0079] 据此,本实施例中,可通过预取的方式,将近期可能被执行的函数对应的函数逻辑数据预先获取下来,这样,一旦这些函数在后续被触发,则可通过半冷启动的方式来执行函数,从而可避免延时更长的冷启动方式。结合前述实施例中描述的方案,可提高采用热启动方式来执行函数的概率,而即使未能采用热启动方式来执行函数,则也可提高采用半冷启动的方式来执行函数的概率,而尽量减少甚至避免使用冷启动的方式来执行函数,因此可有效提高函数的执行效率。

[0080] 需要说明的是,上述实施例所提供方法的各步骤的执行主体均可以是同一设备,或者,该方法也由不同设备作为执行主体。另外,在上述实施例及附图中的描述的一些流程中,包含了按照特定顺序出现的多个操作,但是应该清楚了解,这些操作可以不按照其在本文中出现的顺序来执行或并行执行,操作的序号如100、101等,仅仅是用于区分开各个不同的操作,序号本身不代表任何的执行顺序。另外,这些流程可以包括更多或更少的操作,并且这些操作可以按顺序执行或并行执行。

[0081] 图5为本申请另一示例性实施例提供的一种函数计算方法的流程示意图,参考图5,该方法可包括:

[0082] 步骤500、接收针对目标函数的函数触发请求;

[0083] 步骤501、若目标函数被调度至目标主机上已启动的指定容器,则将指定容器在目标主机上已启动的所有容器中所处的排序位置进行前移,以延长指定容器的驻留时间;

[0084] 步骤502、基于目标函数对应的函数逻辑数据,利用指定容器执行目标函数,目标函数对应的函数逻辑数据预先留存于目标主机上。

[0085] 本实施例提供的函数计算方法可应用于FaaS场景中,以提高FaaS场景中的函数执行效率。

[0086] 本实施例中,可由路由设备来进行函数触发指令的调度。举例来说,路由设备可从容器的维度来进行函数触发指令的调度:路由设备可将指定容器的位置信息配置到函数触发指令中,从而将函数触发指令调度至指定容器,其中位置信息中可包括但不限于容器所处主机的地址等。当然,这是在存在可负载目标函数的容器的情况下的调度方案;而在不存在可负载目标函数的容器的情况下路由设备可将函数触发指令调度至指定主机上,并由指定主机来为目标函数创建新的容器。对目标函数的一种示例性调度原则可以是:优先调度

至已启动的容器;优先调度至上一次负载目标函数的容器上。

[0087] 本实施例中,主机上已启动容器可绑定已经创建/更新过的目标函数对应的函数逻辑数据,基于此,若目标函数被调度至目标主机上的指定容器,则可由指定容器来运行目标函数对应的函数逻辑数据,以执行目标函数;而且,按照上述提及的示例性调度原则,目标函数对应的函数逻辑数据大概率已经留存在目标主机上,因此,目标主机可基于留存的目标函数对应的函数逻辑数据,利用指定容器执行目标函数。

[0088] 而由于指定容器的被请求次数发生了新增,因此,指定容器的排序位置可前移,当然,若其它已启动容器的被请求次数也发生变化,则还是要统筹考虑目标主机上已启动的所有容器,来确定它们之间的排序。本实施例中,可根据指定容器的排序位置来确定指定容器的驻留时间。也即是,可将已启动的容器之间的排序结果,作为确定已启动的容器各自所能获得的驻留时间的依据,排序越靠前的已启动的容器将获得更多的驻留时间。

[0089] 一种可选的实现方案中,可采用前述容器的生命周期管理方法的实施例中提供的方案来确定指定容器的排序位置以及驻留时间。为避免重复,在此不再详述技术细节。

[0090] 本实施例中,还可采用其它实现方案来确定指定容器的排序位置和驻留时间,每被访问一次,排序位置加1,驻留时间同步延迟1分钟,等。

[0091] 本实施例中,用户除了可发起函数触发指令外,还可能发起创建/更新指令。其中,创建指令用于指示创建函数,更新指令则用于指示更新函数。目前,这两个指令均是为了函数执行做准备工作,只有用户随后发起函数触发指令时,才会真正去执行对应的函数。

[0092] 本实施例中,对创建/更新指令进行了创新性的使用,具体地:可在接收到针对目标函数的创建/更新指令的情况下,若创建/更新指令被调度至目标主机上且目标主机上不存在可负载目标函数的容器,则在创建/更新指令的发起方符合指定条件时,控制目标主机预先获取目标函数对应的函数逻辑数据并留存在目标主机上。也即是,在用户发起针对目标函数的创新/更新指令且用户符合指定条件的情况下,可认为目标函数即将发生执行需求,并可通过预取技术,提前将目标函数对应的函数逻辑数据下载到目标主机上,以备用。

[0093] 这里,通过指定条件对预取工作进行了限制,例如,指定条件可以是用户后续发起函数触发指令的可能性符合指定标准。为此,在一种示例性方案中,可根据创建/更新指令的发起方的历史行为记录,统计发起方针对同一函数发起创建/更新指令与发起函数触发指令之间的时间差,以表征该发起方后续发起函数触发指令的可能性,这样,时间差小于指定标准的情况下,才确定该发起方符合指定条件,这种情况下,才执行预取目标函数对应的函数逻辑数据的操作。

[0094] 其中,预取方案可参考前述容器的生命周期管理方案实施例中的描述,在此不再赘述。

[0095] 基于此,在完成上述的预取工作后,后续一旦接收到针对目标函数的函数触发指令,则只需启动新的容器即可,而不再需要远程下载对应的函数逻辑数据,从而,可实现按照前述的半冷启动方式来执行目标函数,这相比于冷启动方式可节省大量的时延。

[0096] 据此,本实施例中,可通过预取的方式,将近期可能被执行的函数对应的函数逻辑数据预先获取下来,这样,一旦这些函数在后续被触发,则可通过半冷启动的方式来执行函数,从而可避免延时更长的冷启动方式。综合来看,可提高采用热启动方式来执行函数的概率,而即使未能采用热启动方式来执行函数,则也可提高采用半冷启动的方式来执行函数

的概率,而尽量减少甚至避免使用冷启动的方式来执行函数,因此可有效提高函数的执行效率。

[0097] 图6为本申请又一示例性实施例提供的一种路由设备的结构示意图。如图6所示,该路由设备包括存储器60、处理器61和通信组件62;

[0098] 存储器60用于存储一条或多条计算机指令;

[0099] 处理器61与存储器60和通信组件62耦合,用于执行一条或多条计算机指令,以用于:

[0100] 通过通信组件62监测目标主机上的已启动容器;

[0101] 根据目标主机上已启动容器各自对应的使用热度,对目标主机上的已启动容器进行排序;

[0102] 根据排序结果,从已启动容器中选择目标容器,容器的排序位置符合预设要求;

[0103] 控制目标容器及目标容器关联的函数逻辑数据保持驻留。

[0104] 在一可选实施例中,处理器61还用于:

[0105] 分别确定各个已启动容器的启动延迟、所占用内存的规格和/或历史驻留时间;

[0106] 其中,对目标主机上的已启动容器进行排序的依据除使用热度外,处理器61还用于启动延迟、所占用内存的规格和/或历史驻留时间。

[0107] 在一可选实施例中,处理器61在对目标主机上的已启动容器进行排序时,用于:

[0108] 计算目标主机上的已启动容器各自对应的优先级;

[0109] 按照优先级对目标主机上的已启动容器进行排序;

[0110] 其中,优先级与使用热度成正比;优先级与启动延迟成正比;优先级与所占内存的规格成反比;优先级与历史驻留时间成正比。

[0111] 在一可选实施例中,处理器61在从已启动容器中选择目标容器时,用于:

[0112] 分别为各个已启动容器构建容器画像;

[0113] 为目标主机维护容器队列;

[0114] 按照排序结果,确定各个已启动容器的容器画像在容器队列中的排序位置;

[0115] 从容器队列中,选择排序位置符合预设要求的容器画像对应的已启动容器,作为目标容器。

[0116] 在一可选实施例中,处理器61在选择排序位置符合预设要求的容器画像对应的已启动容器,作为目标容器时,用于:

[0117] 将位于容器队列之内的容器画像对应的已启动容器,确定为目标容器。

[0118] 在一可选实施例中,处理器61还用于:

[0119] 接收针对目标函数的函数触发指令;

[0120] 若目标函数被调度至目标主机上目标容器中的指定容器,则将指定容器的排序位置前移;

[0121] 基于留存的目标函数对应的函数逻辑数据,利用指定容器执行目标函数。

[0122] 在一可选实施例中,处理器61还用于:

[0123] 在目标用户发起的针对目标函数的创建/更新指令的情况下,若创建/更新指令被调度至目标主机上且目标主机上不存在可负载目标函数的容器,则在目标用户符合指定条件时,控制目标主机预先获取目标函数对应的函数逻辑数据并留存在目标主机上。

[0124] 在一可选实施例中,处理器61还用于:

[0125] 在接收到针对目标函数的函数触发指令时,控制目标主机为目标函数创建容器;

[0126] 基于目标主机上预先获取的目标函数对应的函数逻辑数据,利用为目标函数创建的容器执行目标函数。

[0127] 在一可选实施例中,目标主机与集群中的其它主机处于对等网络中,处理器61在控制目标主机预先获取目标函数对应的函数逻辑数据时,用于:

[0128] 控制目标主机优先从对等网络中的其它主机上获取目标函数对应的函数逻辑数据。

[0129] 在一可选实施例中,处理器61还用于:

[0130] 控制已启动容器中排序位置不符合预设要求的容器释放;

[0131] 删除已释放容器关联的函数逻辑数据。

[0132] 在另一种可能的设计中,还可基于图6所示的路由装置执行一种优化的函数计算方案。对此,处理器61可用于:

[0133] 通过通信组件62接收针对目标函数的函数触发请求;

[0134] 若目标函数被调度至目标主机上已启动的指定容器,则将指定容器在目标主机上已启动的所有容器中所处的排序位置进行前移,以延长指定容器的驻留时间;

[0135] 基于目标函数对应的函数逻辑数据,利用指定容器执行目标函数,目标函数对应的函数逻辑数据预先留存于目标主机上。

[0136] 处理器61还可用于:

[0137] 接收针对目标函数的创建/更新指令;

[0138] 若创建/更新指令被调度至目标主机上且目标主机上不存在可负载目标函数的容器,则在创建/更新指令的发起方符合指定条件时,控制目标主机预先获取目标函数对应的函数逻辑数据并留存在目标主机上。

[0139] 值得说明的是,上述关于路由设备的各实施例中的技术细节,可参考前述的方法实施例中的相关描述,为节省篇幅,在此不再赘述,但这不应造成对本申请保护范围的损失。

[0140] 进一步,如图6所示,该路由设备还包括:电源组件63等其它组件。图6中仅示意性给出部分组件,并不意味着路由设备只包括图6所示组件。

[0141] 相应地,本申请实施例还提供一种存储有计算机程序的计算机可读存储介质,计算机程序被执行时能够实现上述方法实施例中可由路由设备执行的各步骤。

[0142] 其中,图6中的存储器,用于存储计算机程序,并可被配置为存储其它各种数据以支持在计算平台上的操作。这些数据的示例包括用于在计算平台上操作的任何应用程序或方法的指令,联系人数据,电话簿数据,消息,图片,视频等。存储器可以由任何类型的易失性或非易失性存储设备或者它们的组合实现,如静态随机存取存储器(SRAM),电可擦除可编程只读存储器(EEPROM),可擦除可编程只读存储器(EPROM),可编程只读存储器(PROM),只读存储器(ROM),磁存储器,快闪存储器,磁盘或光盘。

[0143] 其中,图6中的通信组件被配置为便于通信组件所在设备和其他设备之间有线或无线方式的通信。通信组件所在设备可以接入基于通信标准的无线网络,如WiFi,2G、3G、4G/LTE、5G等移动通信网络,或它们的组合。在一个示例性实施例中,通信组件经由广播信

道接收来自外部广播管理系统的广播信号或广播相关信息。在一个示例性实施例中,所述通信组件还包括近场通信(NFC)模块,以促进短程通信。例如,在NFC模块可基于射频识别(RFID)技术,红外数据协会(IrDA)技术,超宽带(UWB)技术,蓝牙(BT)技术和其他技术来实现。

[0144] 其中,图6中的电源组件,为电源组件所在设备的各种组件提供电力。电源组件可以包括电源管理系统,一个或多个电源,及其他与为电源组件所在设备生成、管理和分配电力相关联的组件。

[0145] 本领域内的技术人员应明白,本申请的实施例可提供为方法、系统、或计算机程序产品。因此,本申请可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且,本申请可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质(包括但不限于磁盘存储器、CD-ROM、光学存储器等)上实施的计算机程序产品的形式。

[0146] 本申请是参照根据本申请实施例的方法、设备(系统)、和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理设备的处理器以产生一个机器,使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

[0147] 这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中,使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品,该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。

[0148] 这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上,使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理,从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

[0149] 在一个典型的配置中,计算设备包括一个或多个处理器(CPU)、输入/输出接口、网络接口和内存。

[0150] 内存可能包括计算机可读介质中的非永久性存储器,随机存取存储器(RAM)和/或非易失性内存等形式,如只读存储器(ROM)或闪存(flash RAM)。内存是计算机可读介质的示例。

[0151] 计算机可读介质包括永久性和非永久性、可移动和非可移动媒体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。计算机的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,磁带磁磁盘存储或其他磁性存储设备或任何其他非传输介质,可用于存储可以被计算设备访问的信息。按照本文中的界定,计算机可读介质不包括暂存电脑可读媒体(transitory media),如调制的数据信号和载波。

[0152] 还需要说明的是,术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性的包含,从而使得包括一系列要素的过程、方法、商品或者设备不仅包括那些要素,而且还包括没有明确列出的其他要素,或者是还包括为这种过程、方法、商品或者设备所固有的要素。在没有更多限制的情况下,由语句“包括一个……”限定的要素,并不排除在包括所述要素的过程、方法、商品或者设备中还存在另外的相同要素。

[0153] 以上所述仅为本申请的实施例而已,并不用于限制本申请。对于本领域技术人员来说,本申请可以有各种更改和变化。凡在本申请的精神和原理之内所作的任何修改、等同替换、改进等,均应包含在本申请的权利要求范围之内。

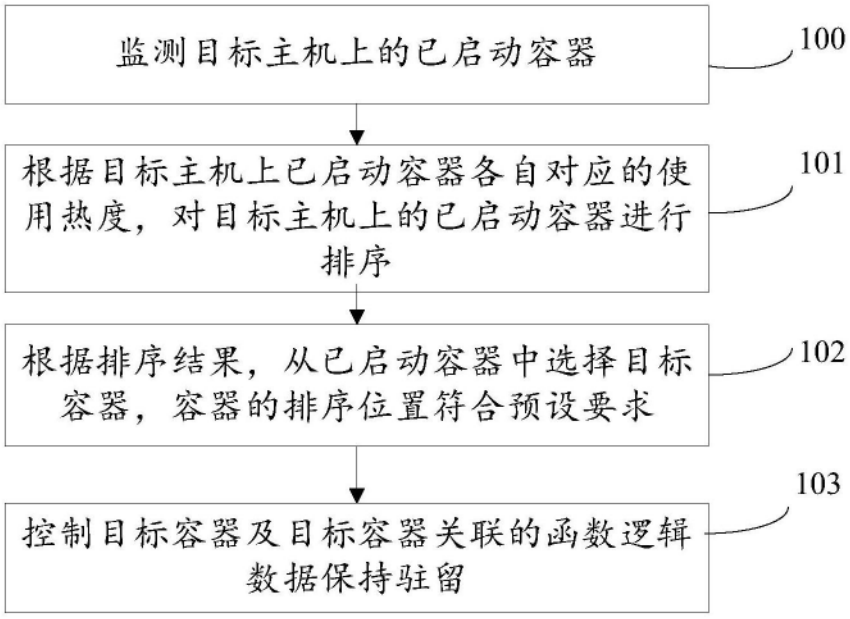

监测目标主机上的已启动容器
100
根据目标主机上已启动容器各自对应的使用热度，对目标主机上的已启动容器进行排序
101
根据排序结果，从已启动容器中选择目标容器，容器的排序位置符合预设要求
102
控制目标容器及目标容器关联的函数逻辑数据保持驻留
103

图1

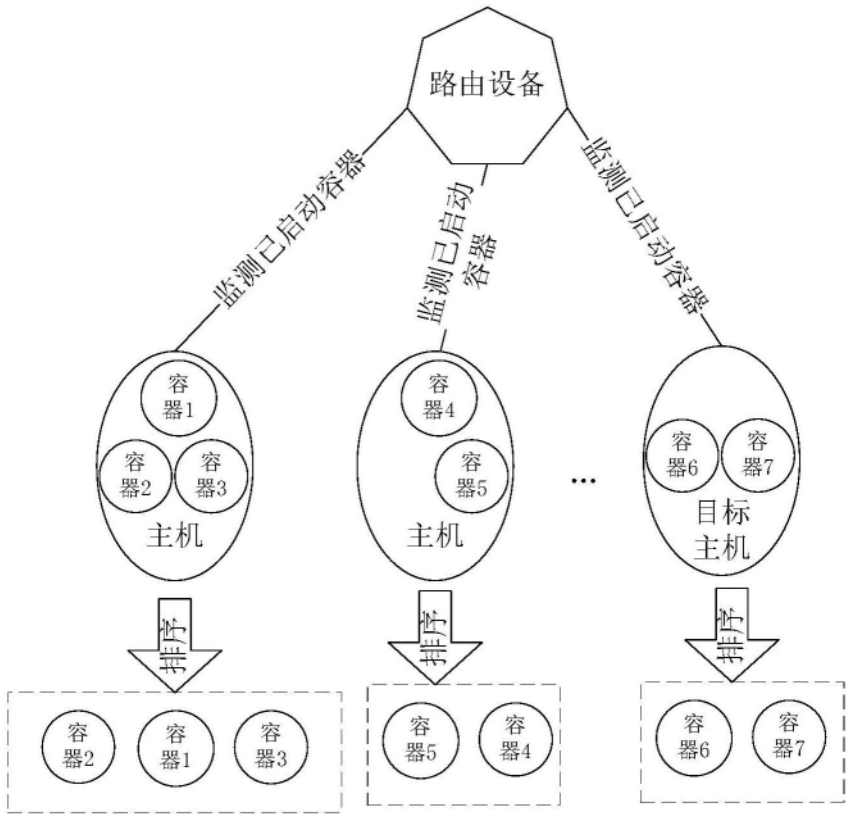

路由设备
监测已启动容器
监测已启动容器
监测已启动容器
容器1
容器2
容器3
主机
容器4
容器5
主机
...
容器6
容器7
目标主机
排序
排序
排序
容器2
容器1
容器3
容器5
容器4
容器6
容器7

图2

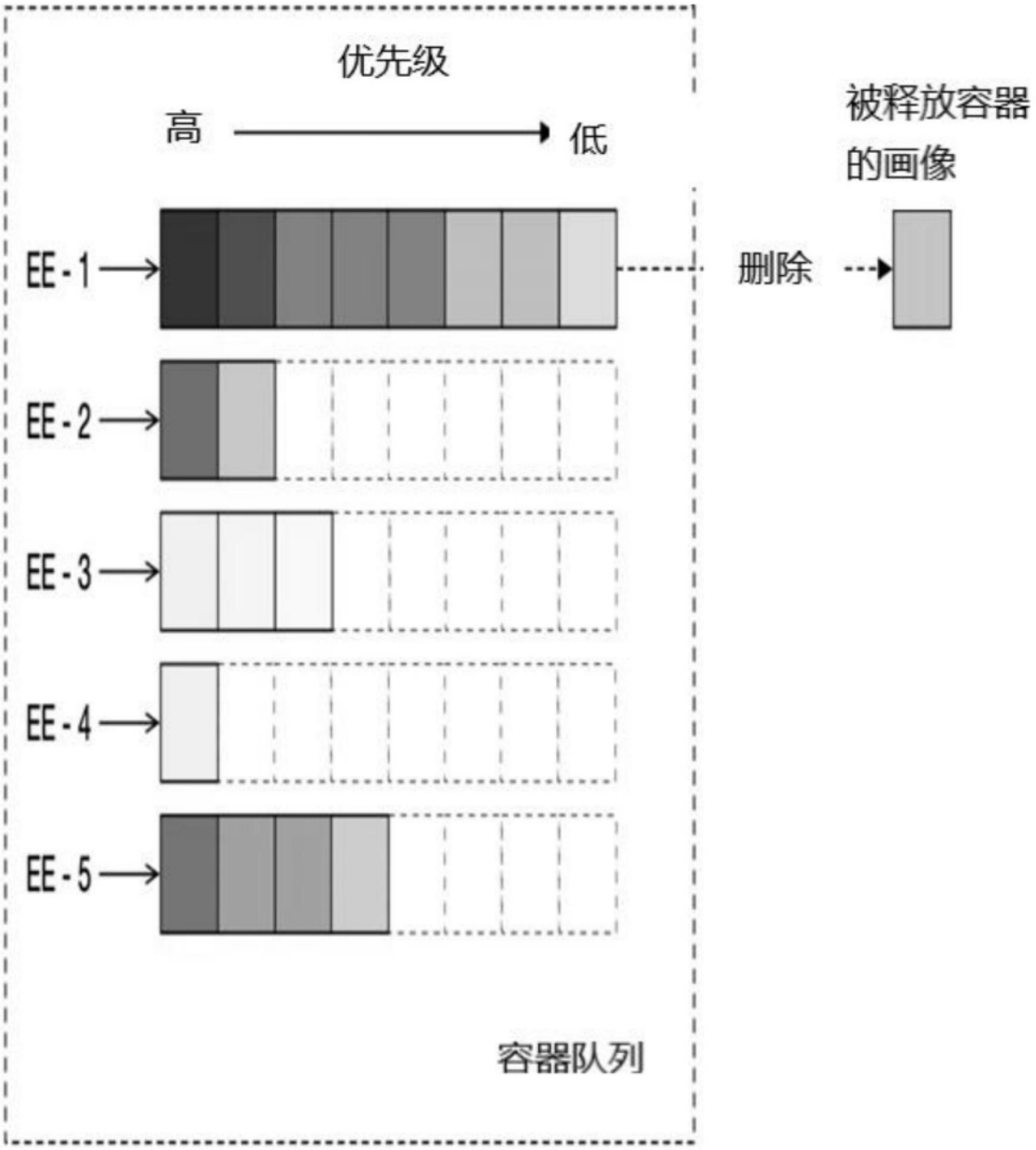


图3

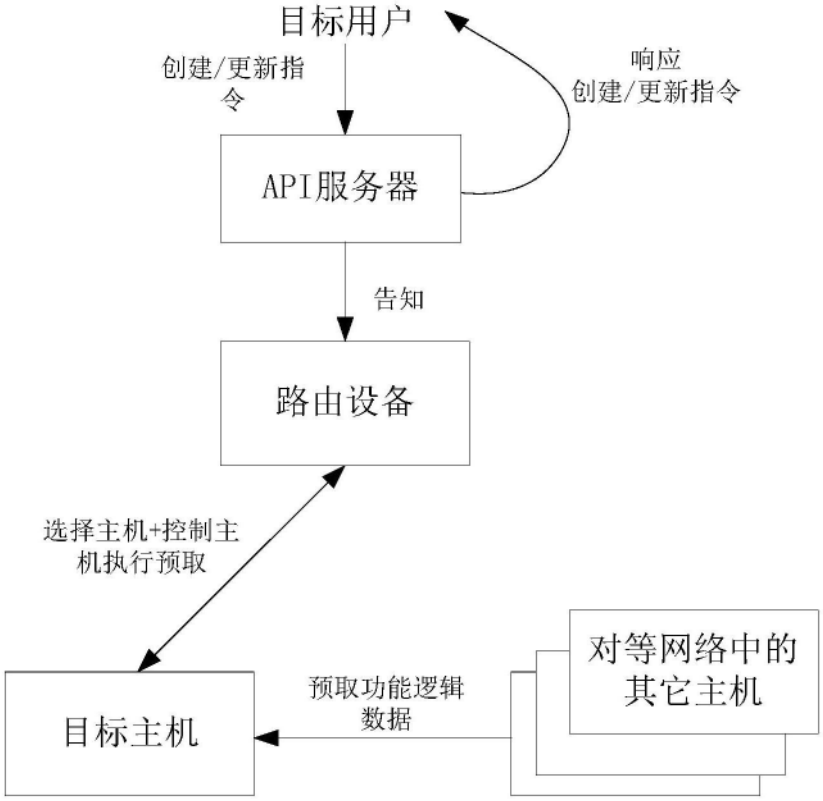

目标用户
创建/更新指令
响应创建/更新指令
API服务器
告知
路由设备
选择主机+控制主机执行预取
目标主机
预取功能逻辑数据
对等网络中的其它主机

图4

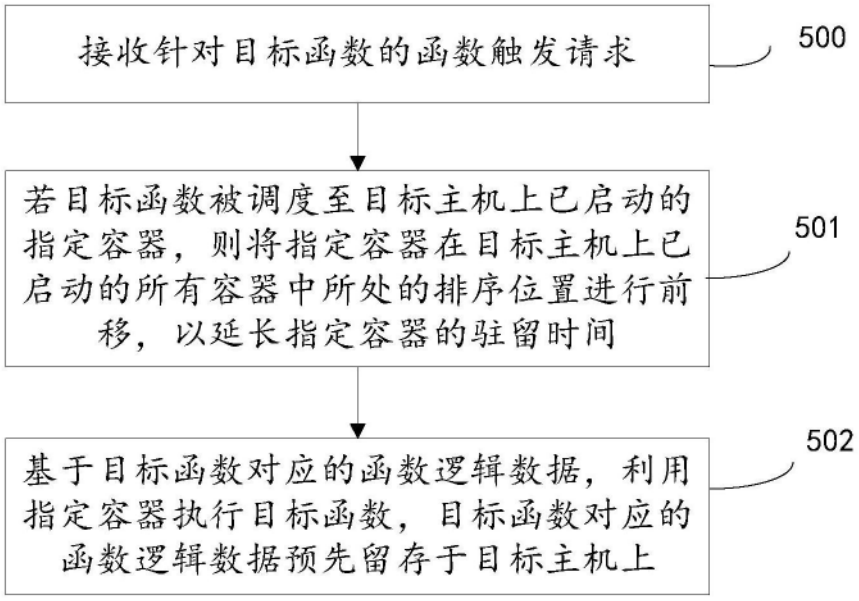

接收针对目标函数的函数触发请求
500
若目标函数被调度至目标主机上已启动的指定容器，则将指定容器在目标主机上已启动的所有容器中所处的排序位置进行前移，以延长指定容器的驻留时间
501
基于目标函数对应的函数逻辑数据，利用指定容器执行目标函数，目标函数对应的函数逻辑数据预先留存于目标主机上
502

图5

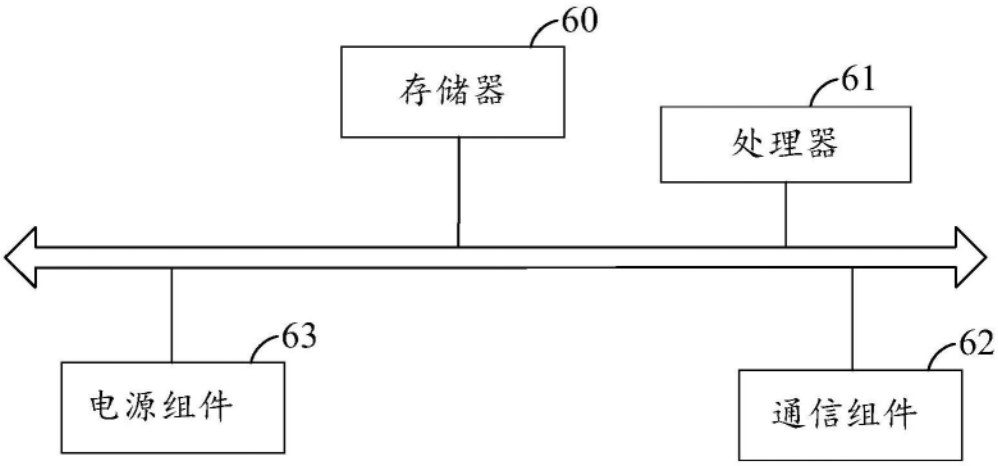


图6