



(12) 发明专利

(10) 授权公告号 CN 112866321 B

(45) 授权公告日 2024. 06. 18

(21) 申请号 201911192927.9

H04L 67/10 (2022.01)

(22) 申请日 2019.11.28

(56) 对比文件

(65) 同一申请的已公布的文献号

CN 109376009 A, 2019.02.22

申请公布号 CN 112866321 A

CN 109936604 A, 2019.06.25

(43) 申请公布日 2021.05.28

审查员 黄欣欣

(73) 专利权人 中兴通讯股份有限公司

地址 518057 广东省深圳市南山区高新技术产业园科技南路中兴通讯大厦法务部

(72) 发明人 唐波 王科文

(74) 专利代理机构 北京天昊联合知识产权代理有限公司 11112

专利代理师 姜春咸 冯建基

(51) Int. Cl.

H04L 67/60 (2022.01)

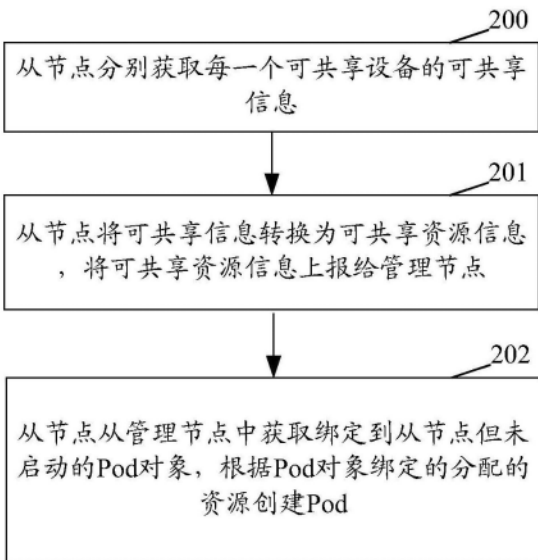
权利要求书2页 说明书14页 附图3页

(54) 发明名称

一种资源调度方法、装置和系统

(57) 摘要

本发明实施例公开了一种资源调度方法、装置和系统,包括:管理节点接收用户的创建Pod请求;其中,创建Pod请求包括:每一个容器对每一种可共享资源的要求;管理节点为待创建的Pod对象选择节点;根据选择的节点中的可共享设备的可共享资源信息为所述Pod对象分配每一个容器的每一种可共享资源,将所述Pod对象、选择的节点和分配的资源进行绑定,存储绑定选择的节点和分配的资源Pod对象。本发明实施例引入了对共享资源的管理调度机制,节约了资源,提高了资源利用率。



1. 一种资源调度方法,包括:

管理节点接收用户的创建Pod请求;其中,所述创建Pod请求包括:每一个容器对每一种可共享资源的要求;所述对每一种可共享资源的要求包括:请求的可共享资源的数量;或者,请求的可共享资源的数量和可共享资源对应的资源属性要求;

管理节点为待创建的Pod对象选择节点;根据选择的节点中的可共享设备的可共享资源信息为所述Pod对象分配每一个容器的每一种可共享资源;其中,所述可共享资源信息包括:设备身份标识ID号、设备的资源属性信息、设备的最大可共享数目。

2. 根据权利要求1所述的方法,其特征在于,该方法还包括:

所述管理节点接收到从节点上报的可共享设备的可共享资源信息,存储或更新所述可共享设备的可共享资源信息。

3. 根据权利要求1所述的方法,其特征在于,其中,所述资源属性要求包括:请求的资源属性值,或资源属性值满足的表达式。

4. 根据权利要求3所述的方法,其特征在于,其中,所述根据选择的节点中的可共享设备的可共享资源信息为Pod对象分配每一个容器的每一种可共享资源包括以下任意一个或多个:

对于每一个容器的每一种可共享资源,当选择的节点存在包括已分配的可共享资源的可共享设备时,为Pod对象分配第一设备中的可共享资源;

对于每一个容器的每一种可共享资源,当选择的节点中的所有可共享设备均未包括已分配的可共享资源时,为Pod对象分配第二设备中的可共享资源;

其中,第一设备包括:

包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目和已分配的可共享资源的数量之差的可共享设备;

或者,包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目和已分配的可共享资源的数量之差,且请求的资源属性量小于或等于请求的可共享资源的数量和设备的最大可共享数目的比值,与资源属性值的乘积的可共享设备;

或者,包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目和已分配的可共享资源的数量之差,且资源属性量满足所述资源属性要求中的表达式的可共享设备;

第二设备包括:

未包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目的可共享设备;

或者,未包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目,且请求的资源属性量小于或等于请求的可共享资源的数量和设备的最大可共享数目的比值,与资源属性值的乘积的可共享设备;

或者,未包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目,且资源属性量满足所述资源属性要求中的表达式的可共享设备。

5. 一种资源调度装置,包括处理器和计算机可读存储介质,所述计算机可读存储介质中存储有指令,其特征在于,当所述指令被所述处理器执行时,实现以下步骤:

接收用户的创建Pod请求;其中,所述创建Pod请求包括:每一个容器对每一种可共享资源的要求;所述对每一种可共享资源的要求包括:请求的可共享资源的数量;或者,请求的可共享资源的数量和可共享资源对应的资源属性要求;

为待创建的Pod对象选择节点;根据选择的节点中的可共享设备的可共享资源信息为所述Pod对象分配每一个容器的每一种可共享资源。

6.一种计算机可读存储介质,其上存储有计算机程序,其特征在于,所述计算机程序被处理器执行时实现以下步骤:

接收用户的创建Pod请求;其中,所述创建Pod请求包括:每一个容器对每一种可共享资源的要求;所述对每一种可共享资源的要求包括:请求的可共享资源的数量;或者,请求的可共享资源的数量和可共享资源对应的资源属性要求;

为待创建的Pod对象选择节点;根据选择的节点中的可共享设备的可共享资源信息为所述Pod对象分配每一个容器的每一种可共享资源。

7.一种资源调度装置,包括:

Pod请求接收模块,用于接收用户的创建Pod请求;其中,所述创建Pod请求包括:每一个容器对每一种可共享资源的要求;所述对每一种可共享资源的要求包括:请求的可共享资源的数量;或者,请求的可共享资源的数量和可共享资源对应的资源属性要求;

Pod对象绑定模块,用于为待创建的Pod对象选择节点,根据选择的节点中的可共享设备的可共享资源信息为所述Pod对象分配每一个容器的每一种可共享资源。

8.一种资源调度系统,包括:

从节点,用于分别获取每一个可共享设备的可共享信息;其中,所述可共享信息包括:设备身份标识ID号、设备的最大可共享数目、资源属性信息和设备状态;所述设备状态包括:设备是否健康、设备是否可用;将所述可共享信息转换为可共享资源信息,将所述可共享资源信息上报给管理节点;其中,所述可共享资源信息包括:设备ID号、资源属性信息、设备的最大可共享数目;

管理节点,用于接收到从节点上报的可共享设备的可共享资源信息,存储或更新可共享设备的可共享资源信息;其中,所述可共享资源信息包括:设备身份标识ID号、资源属性信息、设备的最大可共享数目;接收用户的创建Pod请求;其中,所述创建Pod请求包括以下描述信息:每一个容器对每一种可共享资源的要求;所述对每一种可共享资源的要求包括:请求的可共享资源的数量;或者,请求的可共享资源的数量和可共享资源对应的资源属性要求;为待创建的Pod对象选择节点,根据选择的节点中的可共享设备的可共享资源为所述Pod对象分配每一个容器的每一种可共享资源。

一种资源调度方法、装置和系统

技术领域

[0001] 本发明实施例涉及但不限于计算机应用技术领域,尤指一种资源调度方法、装置和系统。

背景技术

[0002] 以Docker为代表的容器技术越来越被大家所重视,成为云原生(Cloud-Native)应用的重要选择。同时,主流的云平台Kubernetes提供了应用的编排调度、弹缩、高可用(HA, High Available)保证等强大的平台即服务(PaaS, Platform as a Service)化能力,已经是容器调度平台的事实标准。

[0003] Kubernetes在支持CPU、内存(Memory)、存储(Storage)等通用资源管理调度的同时提供了良好的扩展机制,用于扩展外部设备或外部资源(如图形处理单元(GPU, Graphics Processing Unit)、现场可编程门阵列(FPGA, Field Programmable Gate Array)、张量处理单元(TPU, Tensor Processing Unit)等)的管理和调度。用户或者设备厂商通过开发特定的设备插件(Device Plugin)就能很好的支持这些设备在Kubernetes上的调度。但是, Kubernetes现有机制在进行资源调度时只能以设备为单位进行调度。然而,随着集成度的提升,单一设备往往能够提供强大的处理能力。比如GPU的使用,在某些场景下(如人工智能(AI, Artificial Intelligence)模型推理)并不需要一个设备的全部处理能力,所以在此种场景下为单一应用分配一个独占设备时,如果该应用不能使用该设备的全部处理能力,将造成资源的巨大浪费。

发明内容

[0004] 本发明实施例提供了一种资源调度方法、装置和系统,能够节约资源,提高资源利用率。

[0005] 本发明实施例提供了一种资源调度方法,包括:

[0006] 从节点分别获取每一个可共享设备的可共享信息;其中,所述可共享信息包括:设备身份标识ID号、设备的最大可共享数目、资源属性信息和设备状态;所述设备状态包括:设备是否健康、设备是否可用;

[0007] 从节点将所述可共享信息转换为可共享资源信息,将所述可共享资源信息上报给管理节点;其中,所述设备的可共享资源信息包括:设备ID号、资源属性信息、设备的最大可共享数目。

[0008] 本发明实施例提供了一种资源调度方法,包括:

[0009] 管理节点接收用户的创建Pod请求;其中,所述创建Pod请求包括:每一个容器对每一种可共享资源的要求;

[0010] 管理节点为待创建的Pod对象选择节点,根据选择的节点中的可共享设备的可共享资源信息为所述Pod对象分配每一个容器的每一种可共享资源。

[0011] 本发明实施例提供了一种资源调度装置,包括处理器和计算机可读存储介质,所

述计算机可读存储介质中存储有指令,当所述指令被所述处理器执行时,实现上述任一种资源调度方法。

[0012] 本发明实施例提供了一种计算机可读存储介质,其上存储有计算机程序,所述计算机程序被处理器执行时实现上述任一种资源调度方法的步骤。

[0013] 本发明实施例提供了一种资源调度装置,包括:

[0014] 可共享信息获取模块,用于分别获取每一个可共享设备的可共享信息;其中,所述可共享信息包括:设备身份标识ID号、设备的最大可共享数目、资源属性信息和设备状态;所述设备状态包括:设备是否健康、设备是否可用;

[0015] 可共享资源信息上报模块,用于将所述可共享信息转换为可共享资源信息,将所述可共享资源信息上报给管理节点;其中,所述可共享资源信息包括:设备ID号、资源属性信息、设备的最大可共享数目。

[0016] 本发明实施例提供了一种资源调度装置,包括:

[0017] Pod请求接收模块,用于接收用户的创建Pod请求;其中,所述创建Pod请求包括:每一个容器对每一种可共享资源的要求;

[0018] Pod对象绑定模块,用于为待创建的Pod对象选择节点,根据选择的节点中的可共享设备的可共享资源信息为所述Pod对象分配每一个容器的每一种可共享资源。

[0019] 本发明实施例提供了一种资源调度系统,包括:

[0020] 从节点,用于分别获取每一个可共享设备的可共享信息;其中,所述可共享信息包括:设备身份标识ID号、设备的最大可共享数目、资源属性信息和设备状态;所述设备状态包括:设备是否健康、设备是否可用;将所述可共享信息转换为可共享资源信息,将所述可共享资源信息上报给管理节点;其中,所述可共享资源信息包括:设备ID号、资源属性信息、设备的最大可共享数目;

[0021] 管理节点,用于接收到从节点上报的可共享设备的可共享资源信息,存储或更新可共享设备的可共享资源信息;其中,所述可共享资源信息包括:设备身份标识ID号、资源属性信息、设备的最大可共享数目;接收用户的创建Pod请求;其中,所述创建Pod请求包括:每一个容器对每一种资源的要求;为待创建的Pod对象选择节点,根据选择的节点中的可共享设备的可共享资源信息为所述Pod对象分配每一个容器的每一种可共享资源。

[0022] 本发明实施例包括:管理节点接收用户的创建Pod请求;其中,所述创建Pod请求包括:每一个容器对每一种可共享资源的要求;管理节点为待创建的Pod对象选择节点;根据选择的节点中的可共享设备的可共享资源信息为所述Pod对象分配每一个容器的每一种可共享资源。本发明实施例引入了对共享资源的管理调度机制,节约了资源,提高了资源利用率。

[0023] 本发明实施例的其它特征和优点将在随后的说明书中阐述,并且,部分地从说明书中变得显而易见,或者通过实施本发明实施例而了解。本发明实施例的目的和其他优点可通过在说明书、权利要求书以及附图中所特别指出的结构来实现和获得。

附图说明

[0024] 附图用来提供对本发明实施例技术方案的进一步理解,并且构成说明书的一部分,与本发明实施例的实施例一起用于解释本发明实施例的技术方案,并不构成对本发明

实施例技术方案的限制。

- [0025] 图1为相关技术基于Kubernetes对资源的管理和调度的系统架构图;
- [0026] 图2为本发明一个实施例提出的资源调度方法的流程图;
- [0027] 图3为本发明另一个实施例提出的资源调度方法的流程图;
- [0028] 图4为本发明实施例的示例中设备资源共享和独占模式共存的示意图;
- [0029] 图5为本发明另一个实施例提出的资源调度装置的结构组成示意图;
- [0030] 图6为本发明另一个实施例提出的资源调度装置的结构组成示意图;
- [0031] 图7为本发明另一个实施例提出的资源调度系统的结构组成示意图。

具体实施方式

[0032] 下文中将结合附图对本发明实施例进行详细说明。需要说明的是,在不冲突的情况下,本发明中的实施例及实施例中的特征可以相互任意组合。

[0033] 在附图的流程图示出的步骤可以在诸如一组计算机可执行指令的计算机系统中执行。并且,虽然在流程图中示出了逻辑顺序,但是在某些情况下,可以以不同于此处的顺序执行所示出或描述的步骤。

[0034] 图1为目前基于Kubernetes对资源的管理和调度的系统架构图。如图1所示,包括:管理节点(Kube-Master)和至少一个从节点(Kube-Node)。

[0035] 其中,管理节点包括应用服务器(应用程序编程接口(API,Application Programming Interface) Server)和调度程序(Scheduler),每一个从节点包括节点代理(kubelet)和Device plugin。需要说明的是,API Sever和Scheduler可以设置为单独的节点(即管理节点),也可以与kubelet和Device plugin设置在同一个节点上。当设置kubelet和Device plugin的从节点同时设置有API Server和Scheduler时,从节点变为管理节点。

[0036] 下述实施例中以管理节点和从节点进行描述,但是管理节点和从节点可以是同一个节点,也可以是不同的节点。

[0037] 其中,Device plugin,用于获取设备信息,将设备信息上报给kubelet;其中,设备信息包括:设备身份标识(ID,Identifier)号和设备状态;其中,设备状态包括:设备是否健康、是否可用等。

[0038] kubelet,用于将设备信息转换为每一种设备对应的可用的资源数量,将每一种设备对应的可用的资源数量上报给API Server;定时从API Server获取绑定到本节点但未启动的Pod对象,根据Pod对象中的分配的资源信息调用Device Plugin的Allocate接口触发设备的加载并获取配置信息,根据配置信息和分配的资源信息创建Pod。

[0039] API Server,用于存储每一种设备对应的可用的资源数量,接收用户的创建Pod请求;创建Pod请求包括:容器的镜像信息、请求的每一种资源(包括CPU、内存、GPU等)的数量以及Pod的基本信息等;存储绑定节点和可用的资源的Pod对象;

[0040] Scheduler,用于定时从API Server获取待创建的Pod对象,为Pod对象选择每一种可用的资源数量大于或等于请求的资源的数量的节点,从可用的资源中为Pod对象分配资源;将Pod对象、选择的节点和分配的资源进行绑定,即将选择的节点信息和分配的资源信息存储在Pod对象中,将绑定节点和分配的资源Pod对象返回给API Server。

[0041] 在下述实施例中,以第一模块为Device Plugin,第二模块为kubelet,第三模块为

API Server,第四模块为Scheduler来描述。

[0042] 参见图2,本发明一个实施例提出了一种资源调度方法,应用于从节点,包括:

[0043] 步骤200、从节点分别获取每一个可共享设备的可共享信息。

[0044] 具体的,第一模块分别获取每一个可共享设备的可共享信息,将可共享设备的可共享信息上报给第二模块。

[0045] 在本发明实施例中,可共享信息包括:设备ID号、设备的最大可共享数目、资源属性信息和设备状态。

[0046] 在一个示例性实例中,最大可共享数目是指可共享设备作为一种可共享资源被分割得到的所有子资源的数目,具体被分割成多少个子资源可以预先设定。

[0047] 在一个示例性实例中,设备状态包括:设备是否健康、设备是否可用等。

[0048] 在本发明实施例中,可共享设备是指自身能够支持多个容器在设备中同时并发使用的设备,比如GPU、FPGA等。

[0049] 在本发明实施例中,第一模块在将可共享信息上报给第二模块时,如果存在多个可共享设备,则可以将所有可共享设备对应的可共享信息以列表的形式一次性上报给第二模块,也可以一次上报一个可共享设备对应的可共享信息,也可以一次上报多个可共享设备对应的可共享信息。

[0050] 具体在上报每一个可共享设备对应的可共享信息时,可以采用多种方式上报。例如,采用对象、数据结构等方式上报。

[0051] 例如,采用json对象方式上报时,上报的json对象为:

```
ShareableDevice {
```

```
    "ID": "1",    # 设备 ID 号
```

```
    "max_share_num" : 4,    #最大可共享数目
```

[0052]

```
    "properties" : { "memory" : "16GB" },    # 资源属性
```

```
    "health": "Healthy"    # 设备状态
```

```
}
```

[0053] 其中,SharedDevice为设备的对象名,ID为设备ID号,max_share_num为最大可共享数目,properties为资源属性(如GPU显存、设备处理单元数等),health为设备状态。

[0054] 其中,max_share_num和properties为新增字段。

[0055] 在本发明实施例中,第一模块通过ListAndWatch接口将设备的可共享信息上报给第二模块。

[0056] 步骤201、从节点将可共享信息转换为可共享资源信息,将可共享资源信息上报给管理节点。

[0057] 具体的,第二模块将可共享信息转换为可共享资源信息,将可共享资源信息上报给管理节点的第三模块,第三模块存储或更新可共享资源信息。

[0058] 在本发明实施例中,可共享资源信息包括:设备ID号、资源属性信息、设备的最大可共享数目。

[0059] 在本发明实施例中,第二模块在将可共享设备的可共享资源信息上报给第三模块时,如果存在多个可共享设备的可共享资源信息,则可以将可共享设备的可共享资源信息以列表的形式一次性上报给第三模块,也可以一次上报一个可共享设备的可共享资源信息,也可以一次上报多个可共享设备的可共享资源信息。

[0060] 具体在上报每一个可共享设备的可共享资源信息时,可以采用多种方式上报。例如,采用对象、数据结构等方式上报。

[0061] 例如,采用对象方式上报时,上报的对象为:

```
type Resource struct {  
  
    MilliCPU          int64  
  
    Memory            int64  
  
    EphemeralStorage int64  
  
    .....  
[0062] // ScalarResources  
  
    ScalarResources map[v1.ResourceName]int64  
  
    //新增 ShareableResources 字段  
  
    SharableResouces map[v1.ResourceName][]SharableResouce  
  
}
```

[0063] 其中,MilliCPU为CPU毫核数,Memory为内存大小,EphemeralStorage为临时存储大小,ScalarResources为可用的资源数量,SharableResouces为可共享设备的可共享资源信息的列表。

[0064] 其中,可共享设备的可共享资源信息包括:设备ID号、设备的最大可共享数目和资源属性信息。

```
[0065] type SharableResouce struct{  
[0066] Id string  
[0067] Value int64//可分配的共享资源数目  
[0068] Properties map[string][]int  
[0069] }
```

[0070] 其中,Id为设备ID号,Value为可共享设备的最大可共享数目,Properties为资源属性信息。

[0071] 在本发明另一个实施例中,该方法还包括:

[0072] 步骤202、从节点从管理节点中获取绑定到从节点但未启动的Pod对象,根据Pod对象绑定的分配的资源创建Pod。具体的,根据Pod对象绑定的分配的资源触发设备的加载并获取配置信息,根据配置信息和分配的资源信息创建Pod。其中,创建Pod也就是创建Pod中

的所有容器。其中,分配的资源包括可共享资源;或者,可共享资源和非可共享资源。

[0073] 具体的,第二模块从第三模块中获取绑定到从节点但未启动的Pod对象,根据Pod对象绑定的分配的资源调用第一模块触发设备的加载并获取配置信息,根据配置信息和分配的资源信息创建容器。

[0074] 具体的,第二模块根据Pod对象绑定的分配的资源调用第一模块触发设备的加载并获取配置信息包括:

[0075] 第二模块向第一模块发送Allocate接口的请求消息,请求消息包括:分配的可共享资源,分配的可共享资源包括:设备ID号和分配的可共享资源的数量;第一模块向第二模块返回设备ID号对应的可共享设备的配置信息。

[0076] 其中,Allocate接口的请求消息如下:

AllocateRequest [

```
[0077] {  
    "deviceID": 0,  
    "value": 2  
[0078] }
```

[0079] 其中,deviceID为设备ID号,value为分配的可共享资源的数量,单位为slice。当分配的可共享资源的数量与设备的可共享设备的最大可共享数目max_share_num相同时,设备为独占模式;当分配的可共享资源的数量小于可共享设备的最大可共享数目max_share_num时,设备为共享模式。

[0080] 参见图3,本发明另一个实施例提出了一种资源调度方法,应用于管理节点,包括:

[0081] 步骤300、管理节点接收用户的创建Pod请求。

[0082] 具体的,第三模块接收用户的创建Pod请求,根据创建Pod请求创建Pod对象。

[0083] 在本发明实施例中,创建Pod请求包括:每一个容器对每一种可共享资源的要求;其中,对每一种可共享资源的要求包括:可共享资源的数量;或者,可共享资源的数量和可共享资源对应的资源属性要求。

[0084] 其中,资源属性要求包括:请求的资源属性值,或资源属性值满足的表达式。例如,某一资源属性值为10,请求的资源属性值为6;又如,要求资源属性值大于10。

[0085] 在一个示例性实例中,创建Pod请求还包括:容器的镜像信息、以及Pod的基本信息等。

[0086] 在本发明实施例中,创建Pod请求中请求的每一种可共享资源的数量必须为整数,请求的每一种可共享资源的数量未携带slice单位时,其单位默认为设备,按照独占模式处理,这种情况下,需要将请求的每一种可共享资源的数量转换为以slice为单位;请求的每一种可共享资源的数量携带slice单位时,按照共享模式处理,这种情况下,请求的每一种可共享资源的数量只能小于或等于一个可共享设备的最大可共享数目,也就是说不支持同时以独占模式和共享模式使用设备,例如,用户请求4slice,不能在两个设备上各分配2slice。

[0087] 独占模式是指一个设备为一个容器所独占,并且不因系统引入共享模式而使性能

受损;共享模式是指一个设备同时为多个容器并发使用。

[0088] 例如,申请2slice共享资源,并要求某个属性值大于10时,创建Pod请求包括以下信息:

```
apiVersion: v1 # API 版本

kind: Pod # 资源类型

metadata: # 元信息

    name: share-demo # Pod 的名称

spec: # pod 详细描述

    container: # 容器描述

        - name: shared-container-1 # 容器名称

            image: ai.kubernetes.io/inference:2.0 # 容器使用的镜像信息

resources: # 资源描述

    limits: # 资源限制

        cpu: "1" # 限制容器使用 CPU 数为 1

        memory: "500Mi" # 限制容器使用内存大小为 500Mb

        # 本专利新增共享资源的 yaml 描述方式

    alpha.kubernetes.io/share-resource1: # 共享资源名称

        value: 2slice # 申请可共享资源的 2 个切片

        <property_name>: 10 # 申请某种属性值不小于 10
```

[0090] 又如,只申请2slice共享资源时,创建Pod请求包括以下信息:

...#前面的信息不变

[0091]

```
resources:
```

limits: # 资源限制

cpu: "1" # 限制容器使用 CPU 数为 1

memory: "500Mi" # 限制内存大小为 500Mb

alpha.kubernetes.io/share-resource1: 2slice # 申请可共享资源的 2 个切片

又如, 申请独占使用 2 个设备时, 创建 Pod 请求包括以下信息:

... # 前面的信息不变

[0092]

resources:

limits: # 资源限制

cpu: "1" # 限制容器使用 CPU 数为 1

memory: "500Mi" # 限制内存大小为 500Mb

alpha.kubernetes.io/share-resource1: 2 # 申请独占使用 2 个可共享资源设备

[0093] 步骤301、管理节点为待创建的Pod对象选择节点(可以是管理节点,也可以是从节点),根据选择的节点中的可共享设备的可共享资源信息为Pod对象分配每一个容器的每一种可共享资源。

[0094] 具体的,第四模块定时从第三模块获取待创建的Pod对象,为待创建的Pod对象选择对于请求的每一种可共享资源均存在满足请求的可共享资源的要求的设备的节点,根据选择的节点中的可共享设备的可共享资源信息为Pod对象分配请求的每一种可共享资源。

[0095] 在本发明实施例中,选择的节点需要能够为需要创建的所有容器分配所请求的可共享资源,例如,选择的节点至少要满足:对于创建Pod请求中的每一个容器对每一种可共享资源的要求,在选择的节点中均存在满足可共享资源的要求的可共享设备。当然,满足上述条件后,还要保证对于不同容器对不同可共享资源的要求,满足可共享资源的要求的可共享设备在分配可共享资源时不会发生冲突。

[0096] 在一个示例性实例中,满足可共享资源的要求的可共享设备包括:

[0097] 包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目和已分配的可共享资源的数量之差的可共享设备;

[0098] 或者,包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目和已分配的可共享资源的数量之差,且请求的资源属性量小于或等于请求的可共享资源的数量和设备的最大可共享数目的比值,与资源属性值的乘积的可共享设备;

[0099] 或者,包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目和已分配的资源数量之差,且资源属性量满足所述资源属性要求中的表达式的可共享设备;

[0100] 或者,未包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目的可共享设备;

[0101] 或者,未包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目,且请求的可共享资源属性量小于或等于请求的可共享资源的数量和可共享设备的最大可共享数目的比值,与资源属性值的乘积的可共享设备;

[0102] 或者,未包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目,且资源属性量满足所述资源属性要求中的表达式的可共享设备。

[0103] 在本发明实施例中,根据选择的节点中的可共享设备的可共享资源信息为Pod对象分配每一个容器的每一种可共享资源包括以下任意一个或多个:

[0104] 对于每一个容器的每一种可共享资源,当选择的节点存在包括已分配的可共享资源的可共享设备时,为Pod对象分配第一设备中的资源;

[0105] 对于每一个容器的每一种可共享资源,当选择的节点中的所有可共享设备均未包括已分配的可共享资源时,为Pod对象分配第二设备中的资源;

[0106] 其中,第一设备包括:

[0107] 包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目和已分配的可共享资源的数量之差的可共享设备;

[0108] 或者,包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目和已分配的可共享资源的数量之差,且请求的资源属性量小于或等于请求的可共享资源的数量和可共享设备的最大可共享数目的比值,与资源属性值的乘积的可共享设备;

[0109] 或者,包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目和已分配的可共享资源的数量之差,且资源属性量满足所述资源属性要求中的表达式的可共享设备。

[0110] 第二设备包括:

[0111] 未包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目的可共享设备;

[0112] 或者,未包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目,且请求的资源属性量小于或等于请求的可共享资源的数量和设备的最大可共享数目的比值,与资源属性值的乘积的可共享设备;

[0113] 或者,未包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目,且资源属性量满足所述资源属性要求中的表达式的可共享设备。

[0114] 上述为Pod对象分配每一个容器的每一种可共享资源时,分配给同一个容器的同一种可共享资源要么属于同一个可共享设备,要么分配多个可共享设备给同一个容器的同一种可共享资源,并且,优先分配已经分配过可共享资源的可共享设备。

[0115] 在本发明另一个实施例中,该方法还包括:

[0116] 将Pod对象、选择的节点和分配的资源进行绑定,存储绑定选择的节点和分配的资源Pod对象。其中,分配的资源包括:可共享资源;或者,可共享资源和非共享资源。

[0117] 具体的,第四模块将Pod对象、选择的节点和分配的资源进行绑定,将绑定选择的节点和分配的资源Pod对象返回给第三模块,第三模块存储绑定选择的节点和分配的资

源的Pod对象

[0118] 在本发明实施例中,将Pod对象、选择的节点和分配的资源进行绑定是指将选择的节点信息和资源信息存储在Pod对象中。

[0119] 在本发明另一个实施例中,该方法还包括:

[0120] 管理节点接收到从节点上报的可共享设备的可共享资源信息,存储或更新可共享设备的可共享资源信息。

[0121] 具体的,第三模块接收到第二模块上报的可共享设备的可共享资源信息,存储或更新可共享设备的可共享资源信息。

[0122] 在本发明实施例中,可共享资源信息包括:设备ID号、资源属性信息、设备的最大可共享数目。

[0123] 本发明实施例的资源调度方法的有益效果如下:

[0124] 1.为Kubernetes引入了对共享设备/资源的管理调度机制,并提供良好的扩展机制,方便其它共享设备的引入(比如FPGA、TPU等),可以将设备的利用率提升数倍。

[0125] 2.支持可共享设备的独占模式和共享模式在Kubernetes的混合调度,提升用户易用性,降低运维成本。

[0126] 实施例

[0127] 本实施例描述Kubernetes平台在混合模式下的混合调度GPU设备的方法。

[0128] 以GPU作为外部资源为例,通过本专利扩展后的kubernetes集群解决GPU设备的共享与独占分配、调度问题。Kubernetes集群上同时接收训练job和推理job(即上述Pod对象),并根据job请求自适应的分配GPU资源,为独占和共享模式下的GPU分别调度不同的job。

[0129] 实例环境:搭载Ubuntu16.04系统的Kubernetes集群,包含1个管理节点,1个从节点;节点包含4个GTX 1080Ti GPU,CUDA驱动版本9.0,并安装nvidia-docker。

[0130] 具体操作步骤如下:

[0131] 步骤1:部署包含本专利开发和修改过的Kubernetes scheduler,Kubelet, Device-plugin(配置共享GPU设备的设备的最大可共享数目为8)。

[0132] 步骤2:集群部署完毕后,如图4所示,申请创建训练job,yaml描述如下:

```
# 申请 2 个独立 GPU

apiVersion: v1

kind: Pod

metadata:

  name: train-demo

spec:

[0133] container:

  - name: gpu-container-1

    image: ai.zte.io/train:2.0

    resources:

      limits:

        alpha.kubernetes.io/zte-gpu: 2
```

[0134] 步骤3: Pod创建成功后,在节点上通过nvidia-smi命令查看GPU使用情况,发现进程在两个GPU上执行,表明独占模式GPU分配成功。

[0135] 步骤4: 申请创建推理job,yaml描述如下:

```
# 申请 2slice GPU

apiVersion: v1

[0136] kind: Pod

metadata:

  name: inference-demo
```

spec:

container:

- name: gpu-container-1

[0137] image: ai.zte.io/inference:2.0

resources:

limits:

alpha.kubernetes.io/zte-gpu: 2slice

[0138] 步骤5: Pod创建成功后,首先通过kubectl get pods命令查看pod运行情况,确认MPS Server pod已被创建。然后查看推理job pod创建成功,通过nvidia-smi命令来查看GPU进程使用情况如下:1)两个GPU以独占模式运行,执行进程号为训练job pod内进程;2)一个GPU以共享模式运行,执行进程号为nvidia-cuda-mps-server,同时观察此GPU的显存占用率为1/4左右。

[0139] 本架构方案可以在满足多种job类型的基础上,最大程度的提升GPU设备的利用率。而且独占与共享模式的比例可以根据系统的业务情况自适应进行调整,比如需要共享资源的业务量较多时,通过动态增加GPU共享模式的比例,从而进一步提升GPU设备的利用率。

[0140] 本发明另一个实施例提出了一种资源调度装置,包括处理器和计算机可读存储介质,所述计算机可读存储介质中存储有指令,当所述指令被所述处理器执行时,实现上述任一种资源调度方法。

[0141] 本发明另一个实施例提出了一种计算机可读存储介质,其上存储有计算机程序,所述计算机程序被处理器执行时实现上述任一种资源调度方法的步骤。

[0142] 参见图5,本发明另一个实施例提出了一种资源调度装置,包括:

[0143] 可共享信息获取模块501(如上述Device plugin),用于分别获取每一个可共享设备的可共享信息;其中,所述可共享信息包括:设备身份标识ID号、设备的最大可共享数目、资源属性信息和设备状态;所述设备状态包括:设备是否健康、设备是否可用;

[0144] 可共享资源信息上报模块502(如上述kubelet),用于将所述可共享信息转换为可共享资源信息,将所述可共享资源信息上报给管理节点;其中,所述设备的可共享资源信息包括:设备ID号、资源属性信息、设备的最大可共享数目。

[0145] 在本发明另一个实施例中,可共享资源信息上报模块502还用于:

[0146] 定时从管理节点中获取绑定到从节点但未启动的Pod对象,根据Pod对象绑定的分配的资源创建Pod。

[0147] 本发明实施例的从节点的具体实现过程与前述实施例应用于从节点的资源调度方法的具体实现过程相同,这里不再赘述。

[0148] 参见图6,本发明另一个实施例提出了一种资源调度装置,包括:

[0149] Pod对象创建模块602(如上述API Server),用于接收用户的创建Pod请求;其中,

所述创建Pod请求包括:每一个容器对每一种可共享资源的要求;

[0150] Pod对象绑定模块603(如上述Scheduler),用于为待创建的Pod对象选择节点,根据选择的节点中的可共享设备的可共享资源信息为Pod对象分配每一个容器的每一种可共享资源

[0151] 在本发明另一个实施例中,Pod对象绑定模块603还用于:将Pod对象、选择的节点和分配的资源进行绑定,存储绑定选择的节点和分配的资源Pod对象。

[0152] 在本发明另一个实施例中,还包括:

[0153] 可共享资源信息存储更新模块601(如上述API Server),用于接收到从节点上报的可共享设备的可共享资源信息,存储或更新可共享设备的可共享资源信息;其中,所述可共享资源信息包括:设备身份标识ID号、资源属性信息、设备的最大可共享数目;

[0154] 其中,所述对每一种可共享资源的要求包括:可共享资源的数量;或者,可共享资源的数量和可共享资源对应的资源属性要求;所述资源属性要求包括:请求的资源属性值,或资源属性值满足的表达式。

[0155] 在本发明实施例中,Pod对象绑定模块603具体用于采用以下任意一个或多个方式实现所述根据选择的节点中的可共享设备的可共享资源信息为Pod对象分配每一个容器的每一种可共享资源:

[0156] 对于每一个容器的每一种可共享资源,当选择的节点存在包括已分配的可共享资源的可共享设备时,为Pod对象分配第一设备中的可共享资源;

[0157] 对于每一个容器的每一种可共享资源,当选择的节点中的所有可共享设备均未包括已分配的可共享资源时,为Pod对象分配第二设备中的可共享资源;

[0158] 其中,第一设备包括:

[0159] 包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目和已分配的可共享资源的数量之差的可共享设备;

[0160] 或者,包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目和已分配的可共享资源的数量之差,且请求的资源属性量小于或等于请求的资源数量和可共享设备的最大可共享数目的比值,与资源属性值的乘积的可共享设备;

[0161] 或者,包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目和已分配的可共享资源的数量之差,且资源属性量满足所述资源属性要求中的表达式的可共享设备。

[0162] 第二设备包括:

[0163] 未包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目的可共享设备;

[0164] 或者,未包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目,且请求的资源属性量小于或等于请求的可共享资源的数量和设备的最大可共享数目的比值,与资源属性值的乘积的可共享设备;

[0165] 或者,未包括已分配的可共享资源,且请求的可共享资源的数量小于或等于可共享设备的最大可共享数目,且资源属性量满足所述资源属性要求中的表达式的可共享设备。

[0166] 本发明实施例的管理节点的具体实现过程与前述实施例应用于管理节点的资源调度方法的具体实现过程相同,这里不再赘述。

[0167] 参见图7,本发明另一个实施例提出了一种资源调度系统,包括:

[0168] 从节点701,用于分别获取每一个可共享设备的可共享信息;其中,所述可共享信息包括:设备身份标识ID号、设备的最大可共享数目、资源属性信息和设备状态;所述设备状态包括:设备是否健康、设备是否可用;将所述可共享信息转换为可共享资源信息,将所述可共享资源信息上报给管理节点;其中,所述可共享资源信息包括:设备ID号、资源属性信息、设备的最大可共享数目;

[0169] 管理节点702,用于接收到从节点上报的可共享设备的可共享资源信息,存储或更新可共享设备的可共享资源信息;其中,所述可共享资源信息包括:设备身份标识ID号、资源属性信息、设备的最大可共享数目;接收用户的创建Pod请求;其中,所述创建Pod请求包括:每一个容器对每一种可共享资源的要求;为待创建的Pod对象选择节点,根据选择的节点中的可共享设备的可共享资源信息为Pod对象分配每一个容器的每一种可共享资源。

[0170] 上述从节点701和管理节点702的具体实现方式与前述实施例的具体实现过程相同,这里不再赘述。

[0171] 本领域普通技术人员可以理解,上文中所公开方法中的全部或某些步骤、系统、装置中的功能模块/单元可以被实施为软件、固件、硬件及其适当的组合。在硬件实施方式中,在以上描述中提及的功能模块/单元之间的划分不一定对应于物理组件的划分;例如,一个物理组件可以具有多个功能,或者一个功能或步骤可以由若干物理组件合作执行。某些组件或所有组件可以被实施为由处理器,如数字信号处理器或微处理器执行的软件,或者被实施为硬件,或者被实施为集成电路,如专用集成电路。这样的软件可以分布在计算机可读介质上,计算机可读介质可以包括计算机存储介质(或非暂时性介质)和通信介质(或暂时性介质)。如本领域普通技术人员公知的,术语计算机存储介质包括在用于存储信息(诸如计算机可读指令、数据结构、程序模块或其他数据)的任何方法或技术中实施的易失性和非易失性、可移除和不可移除介质。计算机存储介质包括但不限于RAM、ROM、EEPROM、闪存或其他存储器技术、CD-ROM、数字多功能盘(DVD)或其他光盘存储、磁盒、磁带、磁盘存储或其他磁存储装置、或者可以用于存储期望的信息并且可以被计算机访问的任何其他的介质。此外,本领域普通技术人员公知的是,通信介质通常包含计算机可读指令、数据结构、程序模块或者诸如载波或其他传输机制之类的调制数据信号中的其他数据,并且可包括任何信息递送介质。

[0172] 虽然本发明实施例所揭露的实施方式如上,但所述的内容仅为便于理解本发明实施例而采用的实施方式,并非用以限定本发明实施例。任何本发明实施例所属领域内的技术人员,在不脱离本发明实施例所揭露的精神和范围的前提下,可以在实施的形式及细节上进行任何的修改与变化,但本发明实施例的专利保护范围,仍须以所附的权利要求书所界定的范围为准。

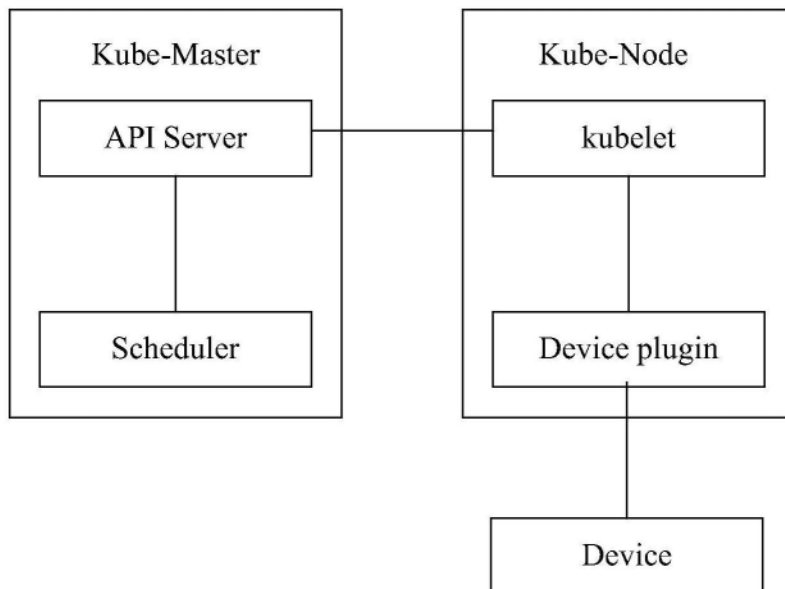


图1

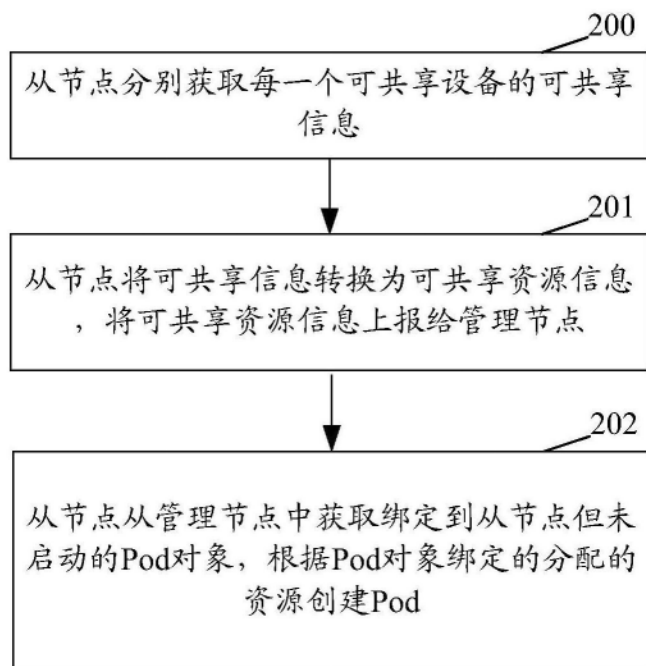


图2

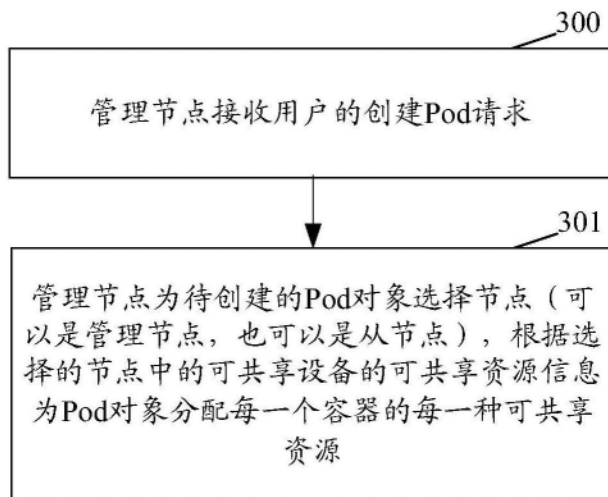


图3

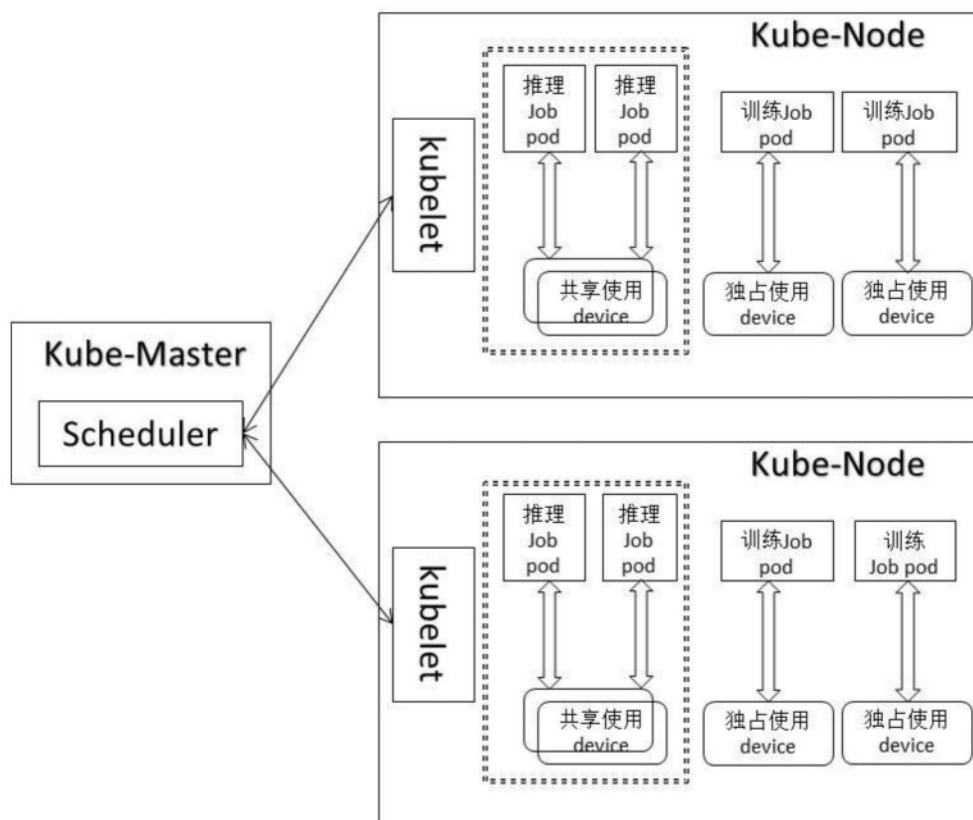


图4



图5

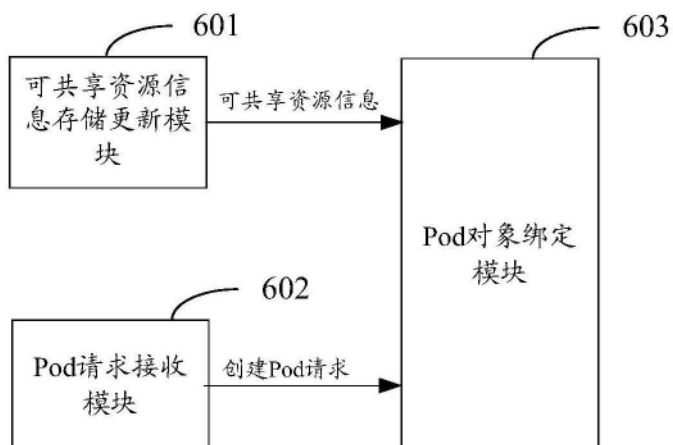


图6

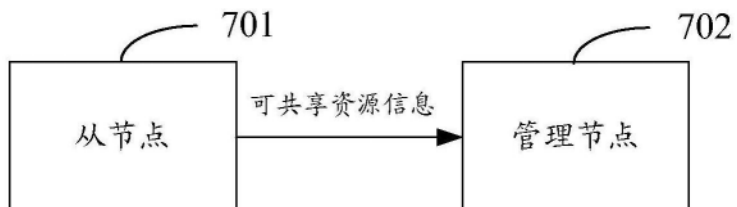


图7