(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 108664116 B
(45)授权公告日 2020.03.27

(21)申请号 201810389653.1

(22)申请日 2018.04.27

(65)同一申请的已公布的文献号
申请公布号 CN 108664116 A

(43)申请公布日 2018.10.16

(73)专利权人 北京邮电大学
地址 100876 北京市海淀区西土城路10号北京邮电大学新科研楼627室

(72)发明人 潘恬 黄韬 杨凡 秦唯特 张娇 刘江 杨帆 谢人超 刘韵洁

(74)专利代理机构 北京清亦华知识产权代理事务所(普通合伙) 11201
代理人 张润

(51)Int.Cl.
G06F 1/3287(2019.01)
G06F 1/329(2019.01)
G06F 9/50(2006.01)

(56)对比文件
CN 107203256 A,2017.09.26,
CN 107203255 A,2017.09.26,
CN 107567704 A,2018.01.09,
CN 107025205 A,2017.08.08,
魏亮、黄韬、张娇等.基于强化学习的服务链映射算法.《通信学报》.2018,第39卷(第1期),

审查员 方源

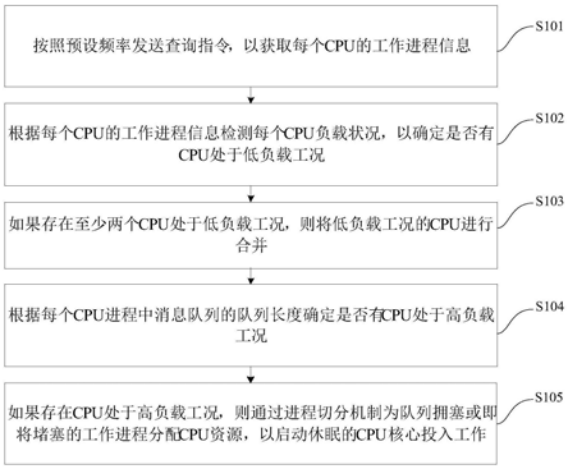
权利要求书2页 说明书10页 附图4页

(54)发明名称

网络功能虚拟化的自适应省电方法、装置及CPU控制器

(57)摘要

本发明公开了一种网络功能虚拟化的自适应省电方法及装置,其中,方法包括以下步骤:按照预设频率发送查询指令,以获取每个CPU的工作进程信息;根据每个CPU的工作进程信息检测每个CPU负载状况,以确定是否有CPU处于低负载工况;如果存在至少两个CPU处于低负载工况,则将低负载工况的CPU进行合并;根据每个CPU进程中消息队列的队列长度确定是否有CPU处于高负载工况;如果存在CPU处于高负载工况,则通过进程切分机制为队列拥塞或即将堵塞的工作进程分配CPU资源,以启动休眠的CPU核心投入工作。该方法可以有效的改善在通用处理器上部署的NFV能效较低的问题,提高NFV能耗,提升系统性能。



CN 108664116 B

1.一种网络功能虚拟化的自适应省电方法,其特征在于,包括以下步骤:

按照预设频率发送查询指令,以获取每个CPU的工作进程信息;

根据所述每个CPU的工作进程信息检测每个CPU负载状况,以确定是否有CPU处于低负载工况;

如果存在至少两个所述CPU处于低负载工况,则将低负载工况的CPU进行合并;

根据每个CPU进程中消息队列的队列长度确定是否有CPU处于高负载工况;以及

如果存在CPU处于高负载工况,则通过进程切分机制为队列拥塞或即将堵塞的工作进程分配CPU资源,以启动休眠的CPU核心投入工作,其中,所述启动休眠的CPU核投入工作,进一步包括:接收检测到数据接收队列发生拥堵生成的告警信息;检测是否存在处于关闭状态的CPU核;如果存在所述处于关闭状态的CPU核,则统计所述告警信息的进程所在的CPU,并通过切分算法切分所述告警信息的进程所在的CPU上的进程,以分为多组进程,并启动所述关闭状态的CPU核,且将切分后的任意一组进程迁移至启动后的所述关闭状态的CPU核。

2.根据权利要求1所述的网络功能虚拟化的自适应省电方法,其特征在于,所述将低负载工况的CPU进行合并,进一步包括:

发送控制信令至任意一个CPU核上的全部进程,以使所述任意一个CPU上的全部进程全部迁移到另一个CPU核上工作;

关闭不再有进程在其上工作的CPU核。

3.根据权利要求1所述的网络功能虚拟化的自适应省电方法,其特征在于,所述通过切分算法切分所述告警信息的进程所在的CPU上的进程,进一步包括:

通过扩展KL算法切分同一个CPU核上的进程,以使得两个集合间的队列权重最小,且所述两个集合的工作强度相同。

4.根据权利要求3所述的网络功能虚拟化的自适应省电方法,其特征在于,所述通过扩展KL算法切分同一个CPU核上的进程,进一步包括:

接收切分请求;

根据所述切分请求所在的CPU将在所述所在的CPU上运行的所有进程的信息做成新的邻接矩阵以记录进程之间的边权重信息,连同端权重数组以记录进程所在CPU物理核的利用率传递给切分函数,执行切分算法。

5.根据权利要求1所述的网络功能虚拟化的自适应省电方法,其特征在于,还包括:

对每个进程设置CPU亲和性,使其只能在目标处理器核心上运行。

6.根据权利要求1所述的网络功能虚拟化的自适应省电方法,其特征在于,通过消息队列进行通信,以将进程映射到一个或多个物理处理器上。

7.一种网络功能虚拟化的自适应省电装置,其特征在于,包括:

获取模块,用于按照预设频率发送查询指令,以获取每个CPU的工作进程信息;

第一确定模块,用于根据所述每个CPU的工作进程信息检测每个CPU负载状况,以确定是否有CPU处于低负载工况;

低负载工况处理模块,用于存在至少两个所述CPU处于低负载工况时,将低负载工况的CPU进行合并;

第二确定模块,用于根据每个CPU进程中消息队列的队列长度确定是否有CPU处于高负载工况;以及

高负载工况处理模块，用于存在CPU处于高负载工况时，通过进程切分机制为队列拥塞或即将堵塞的工作进程分配CPU资源，以启动休眠的CPU核心投入工作，其中，所述启动休眠的CPU核投入工作，进一步包括：接收检测到数据接收队列发生拥堵生成的告警信息；检测是否存在处于关闭状态的CPU核；如果存在所述处于关闭状态的CPU核，则统计所述告警信息的进程所在的CPU，并通过切分算法切分所述告警信息的进程所在的CPU上的进程，以分为多组进程，并启动所述关闭状态的CPU核，且将切分后的任意一组进程迁移至启动后的所述关闭状态的CPU核。

8.根据权利要求7所述的网络功能虚拟化的自适应省电装置，其特征在于，所述低负载工况处理模块进一步地用于发送控制信令至任意一个CPU核上的全部进程，以使所述任意一个CPU上的全部进程全部迁移到另一个CPU核上工作并关闭不再有进程在其上工作的CPU核。

9.一种可感知系统能耗的CPU控制器，其特征在于，包括如权利要求7和8所述网络功能虚拟化的自适应省电装置。

网络功能虚拟化的自适应省电方法、装置及CPU控制器

技术领域

[0001] 本发明涉及计算机网络技术领域，特别涉及一种网络功能虚拟化的自适应省电方法、装置及CPU控制器。

背景技术

[0002] DVFS(Dynamic Voltage And Frequency Scaling，动态电压频率调节)是计算机系统结构中的一种电源管理技术，它可以根据处理器实时的使用状况，对处理器的运行频率和电压进行调整。动态调节芯片的运行频率和电压(对于同一芯片，频率越高，需要的电压也越高)，从而达到节能的目的。

[0003] 尽管DVFS以及电源策略管理器技术在桌面系统及云计算场景被广泛使用，并且已经被内置于大部分的CPU(Central Processing Unit，中央处理器)以及设备驱动中，这种技术并不适合NFV(Network Function Virtualization，网络功能虚拟化)场景。在NFV场景下，相比端到端的任务处理模式，网络内数据包处理对延迟的敏感性较高。使用DVFS或节能策略管理器会在电源状态转换期间引入显著的延迟尖峰(latency spikes)，从而导致数据包处理性能的不确定性。实际上，大多数NFV平台都推荐禁用处理器的C-states和P-states(两种CPU节能模式)来满足数据包的延迟要求。通常在NFV中，CPU的电源状态被固定在x86POLL idle模式执行忙碌循环，以此来为到来的数据包处理任务提供快速响应。

[0004] 然而，CPU的电源状态被固定在x86POLL idle模式执行忙碌循环，以此来为到来的数据包处理任务提供快速响应，但在负载较轻的场景下会引入不必要的能耗。在低负载场景下关闭CPU内核后，如果不能在流量负载上升后，及时重启关闭的CPU核心投入工作，整个系统的性能将大打折扣，CPU资源无法被充分利用。

发明内容

[0005] 本发明旨在至少在一定程度上解决相关技术中的技术问题之一。

[0006] 为此，本发明的一个目的在于提出一种网络功能虚拟化的自适应省电方法，该方法可以有效的改善在通用处理器上部署的NFV能效较低的问题，提高NFV能耗，提升系统性能。

[0007] 本发明的另一个目的在于提出一种网络功能虚拟化的自适应省电装置。

[0008] 本发明的再一个目的在于提出一种可感知系统能耗的CPU控制器。

[0009] 为达到上述目的，本发明一方面实施例提出了一种网络功能虚拟化的自适应省电方法，包括以下步骤：按照预设频率发送查询指令，以获取每个CPU的工作进程信息；根据所述每个CPU的工作进程信息检测每个CPU负载状况，以确定是否有CPU处于低负载工况；如果存在至少两个所述CPU处于低负载工况，则将低负载工况的CPU进行合并；根据每个CPU进程中消息队列的队列长度确定是否有CPU处于高负载工况；如果存在CPU处于高负载工况，则通过进程切分机制为队列拥塞或即将堵塞的工作进程分配CPU资源，以启动休眠的CPU核心投入工作。

[0010] 本发明实施例的网络功能虚拟化的自适应省电方法，通过对CPU资源的细粒度管理，减少在流量负载较轻的场景下，多核处理器的全部核心投入工作造成的不必要的能耗，在流量负载加重时，能够快速启动休眠的CPU核心投入工作，从而有效的改善在通用处理器上部署的NFV能效较低的问题，提高NFV能耗，提升系统性能。

[0011] 另外，根据本发明上述实施例的网络功能虚拟化的自适应省电方法还可以具有以下附加的技术特征：

[0012] 进一步地，在本发明的一个实施例中，所述将低负载工况的CPU进行合并，进一步包括：发送控制信令至任意一个CPU核上的全部进程，以使所述任意一个CPU上的全部进程全部迁移到另一个CPU核上工作；关闭不再有进程在其上工作的CPU核。

[0013] 进一步地，在本发明的一个实施例中，所述启动休眠的CPU核投入工作，进一步包括：接收检测到数据接收队列发生拥堵生成的告警信息；检测是否存在处于关闭状态的CPU核；如果存在所述处于关闭状态的CPU核，则统计所述告警信息的进程所在的CPU，并通过切分算法切分所述告警信息的进程所在的CPU上的进程，以分为多组进程，并启动所述关闭状态的CPU核，且将切分后的任意一组进程迁移至启动后的所述关闭状态的CPU核。

[0014] 进一步地，在本发明的一个实施例中，所述通过切分算法切分所述告警信息的进程所在的CPU上的进程，进一步包括：通过扩展KL算法切分同一个CPU核上的进程，以使得两个集合间的队列权重最小，且所述两个集合的工作强度相同。

[0015] 进一步地，在本发明的一个实施例中，所述通过扩展KL算法切分同一个CPU核上的进程，进一步包括：接收切分请求；根据所述切分请求所在的CPU将在所述所在的CPU上运行的所有进程的信息做成新的邻接矩阵以记录进程之间的边权重信息，连同端权重数组以记录进程所在CPU物理核的利用率传递给切分函数，执行切分算法。

[0016] 进一步地，在本发明的一个实施例中，还包括：对每个进程设置CPU亲和性，使其只能在目标处理器核心上运行。

[0017] 进一步地，在本发明的一个实施例中，通过消息队列进行通信，以将进程映射到一个或多个物理处理器上。

[0018] 为达到上述目的，本发明另一方面实施例提出了一种网络功能虚拟化的自适应省电装置包括：获取模块，用于按照预设频率发送查询指令，以获取每个CPU的工作进程信息；第一确定模块，用于根据所述每个CPU的工作进程信息检测每个CPU负载状况，以确定是否有CPU处于低负载工况；低负载工况处理模块，用于存在至少两个所述CPU处于低负载工况时，将低负载工况的CPU进行合并；第二确定模块，用于根据每个CPU进程中消息队列的队列长度确定是否有CPU处于高负载工况；高负载工况处理模块，用于存在CPU处于高负载工况时，通过进程切分机制为队列拥塞或即将堵塞的工作进程分配CPU资源，以启动休眠的CPU核心投入工作。

[0019] 本发明实施例的网络功能虚拟化的自适应省电装置，通过对CPU资源的细粒度管理，减少在流量负载较轻的场景下，多核处理器的全部核心投入工作造成的不必要的能耗，在流量负载加重时，能够快速启动休眠的CPU核心投入工作，从而有效的改善在通用处理器上部署的NFV能效较低的问题，提高NFV能耗，提升系统性能。

[0020] 另外，根据本发明上述实施例的网络功能虚拟化的自适应省电装置还可以具有以下附加的技术特征：

[0021] 进一步地，在本发明的一个实施例中，所述低负载工况处理模块进一步地用于发送控制信令至任意一个CPU核上的全部进程，以使所述任意一个CPU上的全部进程全部迁移到另一个CPU核上工作并关闭不再有进程在其上工作的CPU核。

[0022] 为达到上述目的，本发明再一方面实施例提出了一种可感知系统能耗的CPU控制器，包括上述的网络功能虚拟化的自适应省电装置。该CPU控制器通过对CPU资源的细粒度管理，减少在流量负载较轻的场景下，多核处理器的全部核心投入工作造成的不必要的能耗，在流量负载加重时，能够快速启动休眠的CPU核心投入工作，从而有效的改善在通用处理器上部署的NFV能效较低的问题，提高NFV能耗，提升系统性能。

[0023] 本发明附加的方面和优点将在下面的描述中部分给出，部分将从下面的描述中变得明显，或通过本发明的实践了解到。

附图说明

[0024] 本发明上述的和/或附加的方面和优点从下面结合附图对实施例的描述中将变得明显和容易理解，其中：

[0025] 图1为根据本发明一个实施例的网络功能虚拟化的自适应省电方法的流程图；

[0026] 图2为根据本发明一个实施例的合并机制生效前的状态的示意图；

[0027] 图3为根据本发明一个实施例的合并机制生效后的状态的示意图；

[0028] 图4为根据本发明一个实施例的切分机制生效前的状态的示意图；

[0029] 图5为根据本发明一个实施例的切分机制生效后的状态的示意图；

[0030] 图6为根据本发明一个实施例的控制器状态转移图；

[0031] 图7为根据本发明一个实施例的进程NFV系统的功能示意图；

[0032] 图8为根据本发明一个实施例的网络功能虚拟化的自适应省电装置的结构示意图。

具体实施方式

[0033] 下面详细描述本发明的实施例，所述实施例的示例在附图中示出，其中自始至终相同或类似的标号表示相同或类似的元件或具有相同或类似功能的元件。下面通过参考附图描述的实施例是示例性的，旨在用于解释本发明，而不能理解为对本发明的限制。

[0034] 在介绍网络功能虚拟化的自适应省电方法、装置及CPU控制器之前，先简单介绍一下相关技术。

[0035] (1)网络功能虚拟化

[0036] NFV即网络功能虚拟化。通过使用x86等通用性硬件以及虚拟化技术，来承载很多功能的软件处理，从而降低网络昂贵的设备成本。NFV可以通过软硬件解耦及功能抽象，使网络设备功能不再依赖于专用硬件，资源可以充分灵活共享，实现新业务的快速开发和部署，并基于实际业务需求进行自动部署、弹性伸缩、故障隔离和自愈等。

[0037] NFV的典型应用是一些CPU密集型功能，并且对网络吞吐量要求不高的情形。主要包括的虚拟化网络功能有：广域网加速器，信令会话控制器，消息路由器，IDS(入侵检测系统)，DPI(深度包检测)，防火墙等。

[0038] 但是，相比于基于专用集成电路(ASIC)和基于网络处理器(NP)的包转发引擎，在

进行高速的数据包包处理时，通用处理器总体上的能效不高，单位能耗所能提供的性能也不高。

[0039] (2)基于ASIC(Application Specific Integrated Circuit，专用集成电路)的包转发引擎

[0040] 构成转发引擎的ASIC是面向固定IP分组处理流程的超大规模集成电路(VLSI)，其采用硬连线结构系统，可为固定功能提供高性能处理，但几乎没有任何灵活性和可扩展性。ASIC对于大批量需求变化少的应用来说，成本上比较经济，比如企业网市场选择ASIC就很理想。在运营商市场，特别是当前IP网络向IP电信网演进过程中，对硬件需求的变化很快，这将导致面向固定需求的ASIC不能满足不断变化的需求。基于ASIC的包转发引擎的特点为：逻辑固定，采用硬连线结构；几乎没有灵活性；非重复设计成本高；产品上市周期长；功耗低；适用于固定需求的运营商或者企业市场。

[0041] (3)基于NP(Network Processor，网络处理器)的包转发引擎

[0042] NP通常将若干微处理器(或称微引擎)内嵌至一个芯片，每个微处理器支持多线程并行处理，这样形成了一个并行处理+流水线的体系。NP还针对包处理进行优化设计，有专门的指令集和配套的软件开发系统。NP具有很强的编程能力，可以完成从2层到7层的多种应用，同时支持新的功能或新的标准的实现，以满足各种各样的网络应用。基于NP的包转发引擎的特点为：集成若干微处理器；灵活性高；非重复设计成本低；产品上市周期短；功耗中；适用于不断发展的运营商市场。

[0043] (4)处理器亲和性

[0044] 通过设置处理器亲和性可以将进程映射到一个或多个物理处理器上。调度算法队列(queue)中的每一个任务(进程或线程)都有一个标签(tag)来指定它们倾向的处理器。在分配处理器的阶段，每个任务就会分配到它们所倾向的处理器上。

[0045] 处理器亲和性利用了这样一个特性，即进程上一次运行后的残余信息会保留在处理器的状态中(也就是指处理器的缓存)。如果下一次仍然将该进程调度到同一个处理器上，就能避免一些不好的情况(比如缓存未命中)，使得进程的运行更加高效。

[0046] 调度算法对于处理器亲和性的支持各不相同。有些调度算法在它认为合适的情况下会允许把一个任务调度到不同的处理器上。比如当两个计算密集型的任务(A和B)同时对一个处理器具有亲和性时，另外一个处理器可能就被闲置了。这种情况下许多调度算法会把任务B调度到第二个处理器上，使得多处理器的利用更加充分。

[0047] 处理器亲和性能够有效地解决一些高速缓存的问题，但却不能缓解负载均衡的问题。

[0048] 正是基于上述原因，本发明实施例提出了一种网络功能虚拟化的自适应省电方法、装置及CPU控制器。

[0049] 下面参照附图描述根据本发明实施例提出的网络功能虚拟化的自适应省电方法、装置及CPU控制器，首先将参照附图描述根据本发明实施例提出的网络功能虚拟化的自适应省电方法。

[0050] 图1是本发明一个实施例的网络功能虚拟化的自适应省电方法的流程图。

[0051] 如图1所示，该网络功能虚拟化的自适应省电方法包括以下步骤：

[0052] 在步骤S101中，按照预设频率发送查询指令，以获取每个CPU的工作进程信息。

[0053] 可以理解的是，控制模块定时地同步下发查询指令给数据层，数据层在收到后会返回自己的工作状态信息(包括CPU利用率等)。

[0054] 具体而言，如图2所示，控制器会定时向所有工作进程发送查询命令，工作进程收到后会向控制器上报本进程的已处理数据包个数，队列占用程度，所在物理核编号(cpu)，进程号(pid)等信息，控制器收到后会更新对应进程的记录内容，当收齐所有进程的信息后会更新每个进程的CPU使用率(cpuUsage)。

[0055] 在步骤S102中，根据每个CPU的工作进程信息检测每个CPU负载状况，以确定是否有CPU处于低负载工况。

[0056] 可以理解的是，控制模块收到返回信息后，会检查所有CPU的负载状况，并决定是否有CPU处于低负载工况。如果CPU不处于低负载工况，则系统不做处理。

[0057] 也就是说，控制模块按照预设频率不断发出查询指令，并根据反馈的信息(至少有两个可合并的CPU)判断是否处于低负载工况。

[0058] 在步骤S103中，如果存在至少两个CPU处于低负载工况，则将低负载工况的CPU进行合并。

[0059] 可以理解的是，如果有多于两个以上CPU处于低负载工况，则合并其中的两个CPU(将一个CPU上的所有进程转移到另一个CPU上进行工作)。

[0060] 具体而言，如果CPU物理核使用率低于门限值，便认为该CPU物理核为空闲CPU物理核。然后控制器对所有正在工作(未关闭)的CPU物理核进行检查，如果至少存在两个空闲CPU核，就将二者进行合并。

[0061] 在本发明的一个实施例中，将低负载工况的CPU进行合并，进一步包括：发送控制信令至任意一个CPU核上的全部进程，以使任意一个CPU上的全部进程全部迁移到另一个CPU核上工作；关闭不再有进程在其上工作的CPU核。

[0062] 具体而言，如图3所示，控制器将发送控制信令到其中一个CPU上的全部进程(如图3，Step2)，让这些进程全部迁移到另一个CPU核上工作(如图3，原本工作在3号CPU上的全部进程被迁移到了1号CPU上)，并关闭不再有进程在其上工作的CPU(如图3，3号CPU被关闭)。

[0063] 在步骤S104中，根据每个CPU进程中消息队列的队列长度确定是否有CPU处于高负载工况。

[0064] 可以理解的是，每个工作进程会自己检查其读取的消息队列的队列长度，并以此为基础判断本进程是否处于高负载工况。如果不处于高负载工况，则不做任何处理。

[0065] 在步骤S105中，如果存在CPU处于高负载工况，则通过进程切分机制为队列拥塞或即将堵塞的工作进程分配CPU资源，以启动休眠的CPU核心投入工作。

[0066] 可以理解的是，在控制模块收到来自进程的高负载告警时，会得知系统处于高负载工况，并执行相应处理。如果处于高负载工况，则向控制器异步地发送告警信息。控制器在收到告警信息后会根据已经收集的所有信息，对发出告警信息的进程所在的CPU进行切分，得到切分结果后，再通知相应的进程切换工作CPU。

[0067] 需要说明的是，高负载工况由普通进程自身检测得出后，通知控制模块，控制模块再进行处理。另外，当不属于低负载与高负载工况时(正常工况，无需调整)，或处于高工况而没有空闲CPU时(无法调整)，系统不做处理。

[0068] 具体而言，当系统中的流量状态或CPU工作状态发生变化，导致某一个NFV工作进

程分配到的CPU资源不足时，工作进程处理报文的速度不够快，处理报文速度就有可能低于报文流入进程的速度，这将会导致工作进程的数据接收队列出现拥堵，数据包在队列内排队等候处理，进而导致性能下降。本发明实施例使用进程切分机制为队列拥塞(或即将拥塞)的工作进程分配更多的CPU资源，缓解其压力，提升系统性能。

[0069] 如图4所示，当工作进程(如图4，CPU1)检测到数据接收队列发生拥堵(或即将发生拥堵)时，就会向控制器发送告警信息(urgent message)，请求分配CPU(如图4，Step1)。

[0070] 进一步地，在本发明的一个实施例中，启动休眠的CPU核投入工作，进一步包括：接收检测到数据接收队列发生拥堵生成的告警信息；检测是否存在处于关闭状态的CPU核；如果存在处于关闭状态的CPU核，则统计告警信息的进程所在的CPU，并通过切分算法切分告警信息的进程所在的CPU上的进程，以分为多组进程，并启动关闭状态的CPU核，且将切分后的任意一组进程迁移至启动后的关闭状态的CPU核。

[0071] 可以理解的是，控制器首先检查是否还有处在关闭状态的CPU，如果没有，说明现在所有的可用CPU都在进行工作，没有更多CPU资源可供分配，故不做处理。如果检查发现系统中存在处于关闭状态的CPU物理核(如图4，CPU3)，控制器便对发出告警信息的进程所在的CPU进行统计，并对所有工作在这个CPU上的进程使用切分算法，并将这些进程分为两组(如图5，P1、P2、P3为一组，P4、P5、P6为一组)，启动关闭状态的CPU(如图5，CPU3)并将其中的一组进程迁移到此CPU核上进行工作(如图5，P4、P5、P6被迁移到了CPU3上运行)。

[0072] 进一步地，在本发明的一个实施例中，通过切分算法切分告警信息的进程所在的CPU上的进程，进一步包括：通过扩展KL算法切分同一个CPU核上的进程，以使得两个集合间的队列权重最小，且两个集合的工作强度相同。

[0073] 可以理解的是，在进程切分机制中，使用一种启发式算法——扩展KL算法，实现对在同一个CPU核心上运行的进程的均分，并使得切分后两个集合间的队列权重尽量小(连接两个进程集合的边的队列权重尽量小，即两个集合间的边权重尽量小)，两边的工作强度尽量接近(两个进程集合的CPU利用率尽量接近，即端权重之和尽量接近)。

[0074] 进一步地，在本发明的一个实施例中，通过扩展KL算法切分同一个CPU核上的进程，进一步包括：接收切分请求；根据切分请求所在的CPU将在所在的CPU上运行的所有进程的信息做成新的邻接矩阵以记录进程之间的边权重信息，连同端权重数组以记录进程所在CPU物理核的利用率传递给切分函数，执行切分算法。

[0075] 可以理解的是，控制器在收到某个进程的切分请求后，会根据其所在的CPU，将在此CPU上运行的所有进程的信息做成新的邻接矩阵(记录进程之间的边权重信息)，连同端权重数组(记录进程所在CPU物理核的利用率)传递给切分函数，执行切分算法。

[0076] 具体而言，在同一个CPU物理核上运行的所有进程以及它们用来在数据层进行通信的消息队列可以构成一张图G(V,E)，对一张图G(V,E)来说，V为其所有的端点，对应所有的进程，端权重对应进程的CPU利用率，E为其所有的边，对应所有进程之间的消息队列，$edge_cost_{ij}$为顶点i、j之间边的边权重，对应顶点i、j之间消息队列通过的流量占整个系统拓扑通过流量的比重。已知图G中有2n个端点，将所有顶点V分为两个集合A与B。定义I_{a_i}为在A中的顶点a_i的内部边开销，E_{a_i}为在A中的顶点a_i的外部边开销，D_a为在A中的顶点a的边权重开销差。对在B中的顶点b也有同样的定义。

[0077] $$I_{a_i} = \sum_{a_j \neq a_i} edge\_cost_{a_j a_i}$$

[0078] $$E_{a_i} = \sum_{bj} edge\_cost_{b_j a_i}$$

[0079] $D_a = E_a - I_a$

[0080] 在交换A、B中的顶点a和b之后，整个图的边权重增益定义为gain_edgecut：

[0081] $gain\_edgecut = D_a + D_b - 2edge\_cost_{a,b}$

[0082] 在交换A、B中的顶点a和b之后，整个图的端权重开销差定义为workload_diff：

[0083] $$workload\_diff = abs\left(\sum_{a\ in\ A} pointWeight_a - \sum_{b\ in\ B} pointWeight_b \right)$$

[0084] abs()为求绝对值运算，整个图的端权重差增益定义为gain_workload_diff：

[0085] gain_workload_diff＝workload_diff_old-workload_diff_new

[0086] 扩展KL算法分两部分，一部分考虑端权重，一部分考虑边权重，最后两者加权求和，取收益为正的一对顶点互换。

[0087] 先把待切分的所有端点分为A、B两组，数量相同(或在待切分端点总数量为奇数时相差1)，对两组中的每一个点，计算其内部边开销以及外部边开销并计算边权重增益。之后对A、B两组内的每一对点a与b(a在A中，b在B中)计算两个点互换后图的边权重增益并在边权重增益矩阵中进行记录。

[0088] 端权重算法是类似的，先计算两组端点A、B的端权重开销差，然后计算两组端点里的每一对点a与b(a在A中，b在B中)互换后的端权重差增益并在端权重差增益矩阵中进行记录。

[0089] 将上述两个矩阵中记录的边权重增益与端权重差增益加权求和后得到总增益，总增益为正就互换两个顶点，得到新的分组A’、B’，并对新的分组重新计算边权重增益矩阵与端权重差增益矩阵。循环上述过程直到没有互换增益为止(或达到循环次数上限)。最后得到了两组端点，使得根据这种分组切分同一个CPU物理核上的进程后，两个集合间的队列权重尽量小，两边的工作强度尽量接近。

[0090] 如图6所示，分析切分算法的工作流程：

[0091] (1)从正常状态(Usual)开始，如果控制器收到来自进程的告警信息(urgentmessage)，说明目前有进程分配到的CPU时钟不足，需要控制器为其分配更多CPU资源，于是控制器检查CPU分配情况(CheckCPU)；

[0092] (2)如果检查发现所有的CPU物理核都已分配了工作，说明系统已无力为发出告警信息的进程分配更多CPU资源，CPU资源已经被充分利用，于是返回正常状态(Usual)；如果检查发现存在空闲的CPU物理核，则对发出告警信息之进程所在的CPU物理核上的所有进程使用切分算法(KL)

[0093] (3)得出KL算法计算出的结果后，启动空闲的CPU物理核，并根据切分算法得出的两个分组，将其中一个分组所对应的进程迁移至空闲CPU物理核上运行，控制器也回到正常状态。

[0094] 进一步地，在本发明的一个实施例中，本发明实施例的方法还包括：对每个进程设置CPU亲和性，使其只能在目标处理器核心上运行。

[0095] 可以理解的是，将整个系统划分为数据平面与控制平面；控制信令的传送预留了控制信道，使其与数据信道相隔离；为所有进程设置了CPU亲和性，使其只能在绑定的CPU物理核上运行。

[0096] 进一步地，在本发明的一个实施例中，通过消息队列进行通信，以将进程映射到一个或多个物理处理器上。

[0097] 可以理解的是，控制器能够实时探测到系统目前的流量负载情况以及各工作进程所在CPU的工作情况；在流量负载较轻，时可以通过迁移部分进程，关闭空闲CPU物理核来进行省电；在流量负载较重时，可以将负载过重的CPU上的进程迁移至空闲物理核上，提升系统性能，缓解负载过重CPU物理核的压力。

[0098] 进一步地，如图7所示，本发明实施例进行设计实现的基础是使用消息队列进行通信的多进程NFV系统，系统架构主要分为控制层和数据层，控制层有一个控制进程(即控制器)，控制器与所有数据层上的工作进程维持一个可以双向通信的非阻塞消息队列作为控制信道，用于传输控制信令。数据层的工作进程间则维持单向传输的阻塞消息队列作为数据信道，用于工作进程间数据(多为已处理或待处理的数据包)的传输。为了减少进程在处理器之间频繁迁移产生的开销，也为了提高CPU缓存的命中率以提升性能，本发明实施例为每个进程都设置了CPU亲和性，使其只能在规定的处理器核心上运行，在提升性能的同时也方便控制器对其调度。

[0099] 进一步地，如图6所示，基于上述系统及机制的设计(使用消息队列进行通信的多进程NFV系统、低流量负载下的进程合并机制和高流量负载下的进程切分机制)。分析合并算法的工作流程：

[0100] (1)从正常状态(Usual)开始，控制器定期向所有包处理系统中的工作进程发送询问信息，同时接收工作进程回执的应答信息，并根据收到的应答信息更新边权重信息(UpdateEdgeRecords)；

[0101] (2)一旦发现所有的边权重都更新完毕，便根据收到的信息，更新CPU核心的使用情况，并检查是否存在可以合并的空闲CPU物理核，如果空闲物理核少于两个，则无法进行合并，回到正常状态(Usual)；如至少存在两个可供合并的物理核，则对两个CPU物理核进行合并(Tellprocesses to change CPU and wait for a moment)；

[0102] (3)向待合并的两个CPU物理核(如A与B)其中一个(如B)上的全部进程发送迁移消息，将这些进程迁移到两个待合并CPU物理核中的另一个(如A)上，然后等待迁移完成后，控制器回到正常状态。

[0103] 需要说明的是，对于如交换机和路由器这样的网络设备来说，可以通过将流量聚合成更少的链路并将空闲链路(例如，路由器的线路卡)，置于休眠状态来实现节能。这种休眠机制需要依靠集中式控制器从分布式网络节点收集链路状态，并周期性地做出相应的休眠/唤醒决策来实现。本发明实施例通过充分利用多核处理器的灵活性，把这个来源于流量感知转发的设计模式应用到NFV。为了提高NFV的能效，在一个细粒度控制器的调度下，使得多个NF功能能够在多个CPU核心间迁移。与协调线路卡休眠的模式类似，控制器将不断尝试将NFV功能进程聚合在尽量少的CPU核心上，并使以此产生的空闲CPU内核进入休眠模式以节省功耗。

[0104] 综上，本发明实施例基于运行在多核x86处理器上，使用消息队列进行通信的多进

程IP数据包处理系统进行设计实现。主要设计了使用消息队列进行通信的多进程NFV系统、低流量负载下的进程合并机制和高流量负载下的进程分散机制。多进程NFV系统是整个方案的基础组成与载体，用户可在系统中部署多个不同的NFV模块以进行IP数据包处理；进程合并机制实现了在低流量负载时，在几个负载较轻CPU物理核心上运行的进程能够迁移到其中一个或几个更少的CPU物理核心上运行，同时关闭没有进程运行的物理核心，实现节能；进程切分机制实现在流量负载提升时，进程发现本进程分配到的CPU时钟资源不足，申请系统为其分配更多的CPU资源，控制器重启空闲CPU并为其分配NFV工作进程，从而提升系统性能。

[0105] 另外，本发明实施例的方法带来的有益效果，具体包括：

[0106] (1)使用消息队列进行通信的多进程NFV系统

[0107] a、传输速率快：本发明中的多个NFV进程使用消息队列进行数据传递，封装更少，效率更高。在同一台物理机内，直接在消息队列中传送待处理的IP报文，能够达到很高的传输速率。

[0108] b、NFV功能模块易开发：本发明中的每个NFV模块运行在独立的进程中，每个模块只需要对到来的数据包以及控制器发来的控制消息做出响应，方便功能模块开发。

[0109] (2)低流量负载下的进程合并机制

[0110] 低流量负载下省电：在低流量负载下，本发明可以通过合并算法，在尽量不降低系统性能的前提下，将负载较轻的任务迁移至更少的CPU物理核上，并关闭原所在CPU物理核，达到省电的效果。

[0111] (3)高流量负载下的进程切分机制

[0112] a、在搞流量负载下提升性能：在高流量负载下，本发明可以通过切分算法，将负担较重的CPU物理核上运行的进程部分迁移至空闲的CPU物理核上，提升系统性能。

[0113] b、使用扩展KL算法进行进程切分：使用扩展KL算法切分同一个CPU物理核上的进程后，使得两个集合间的队列权重尽量小，两边的工作强度尽量接近。队列权重小，可以使得跨CPU物理核的数据传输更少，提升系统性能和能效。

[0114] 根据本发明实施例提出的网络功能虚拟化的自适应省电方法，通过对CPU资源的细粒度管理，减少在流量负载较轻的场景下，多核处理器的全部核心投入工作造成的不必要的能耗，在流量负载加重时，能够快速启动休眠的CPU核心投入工作，从而有效的改善在通用处理器上部署的NFV能效较低的问题，提高NFV能耗，提升系统性能。

[0115] 其次参照附图描述根据本发明实施例提出的网络功能虚拟化的自适应省电装置。

[0116] 图8是本发明一个实施例的网络功能虚拟化的自适应省电装置的结构示意图。

[0117] 如图8所示，该网络功能虚拟化的自适应省电装置10包括：获取模块100、第一确定模块200、低负载工况处理模块300、第二确定模块400和高负载工况处理模块500。

[0118] 其中，获取模块100用于按照预设频率发送查询指令，以获取每个CPU的工作进程信息。第一确定模块200用于根据每个CPU的工作进程信息检测每个CPU负载状况，以确定是否有CPU处于低负载工况。低负载工况处理模块300用于存在至少两个CPU处于低负载工况时，将低负载工况的CPU进行合并。第二确定模块400用于根据每个CPU进程中消息队列的队列长度确定是否有CPU处于高负载工况。高负载工况处理模块500用于存在CPU处于高负载工况时，通过进程切分机制为队列拥塞或即将堵塞的工作进程分配CPU资源，以启动休眠的

CPU核心投入工作。本发明实施的装置10可以有效的改善在通用处理器上部署的NFV能效较低的问题，提高NFV能耗，提升系统性能。

[0119] 进一步地，在本发明的一个实施例中，低负载工况处理模块300进一步地用于发送控制信令至任意一个CPU核上的全部进程，以使任意一个CPU上的全部进程全部迁移到另一个CPU核上工作并关闭不再有进程在其上工作的CPU核。

[0120] 需要说明的是，前述对网络功能虚拟化的自适应省电方法实施例的解释说明也适用于该实施例的网络功能虚拟化的自适应省电装置，此处不再赘述。

[0121] 根据本发明实施例提出的网络功能虚拟化的自适应省电装置，通过对CPU资源的细粒度管理，减少在流量负载较轻的场景下，多核处理器的全部核心投入工作造成的不必要的能耗，在流量负载加重时，能够快速启动休眠的CPU核心投入工作，从而有效的改善在通用处理器上部署的NFV能效较低的问题，提高NFV能耗，提升系统性能。

[0122] 此外，本发明实施例提出了一种可感知系统能耗的CPU控制器，包括上述的网络功能虚拟化的自适应省电装置。该CPU控制器通过对CPU资源的细粒度管理，减少在流量负载较轻的场景下，多核处理器的全部核心投入工作造成的不必要的能耗，在流量负载加重时，能够快速启动休眠的CPU核心投入工作，从而有效的改善在通用处理器上部署的NFV能效较低的问题，提高NFV能耗，提升系统性能。

[0123] 此外，术语“第一”、“第二”仅用于描述目的，而不能理解为指示或暗示相对重要性或者隐含指明所指示的技术特征的数量。由此，限定有“第一”、“第二”的特征可以明示或者隐含地包括至少一个该特征。在本发明的描述中，“多个”的含义是至少两个，例如两个，三个等，除非另有明确具体的限定。

[0124] 在本说明书的描述中，参考术语“一个实施例”、“一些实施例”、“示例”、“具体示例”、或“一些示例”等的描述意指结合该实施例或示例描述的具体特征、结构、材料或者特点包含于本发明的至少一个实施例或示例中。在本说明书中，对上述术语的示意性表述不必须针对的是相同的实施例或示例。而且，描述的具体特征、结构、材料或者特点可以在任一个或多个实施例或示例中以合适的方式结合。此外，在不相互矛盾的情况下，本领域的技术人员可以将本说明书中描述的不同实施例或示例以及不同实施例或示例的特征进行结合和组合。

[0125] 尽管上面已经示出和描述了本发明的实施例，可以理解的是，上述实施例是示例性的，不能理解为对本发明的限制，本领域的普通技术人员在本发明的范围内可以对上述实施例进行变化、修改、替换和变型。

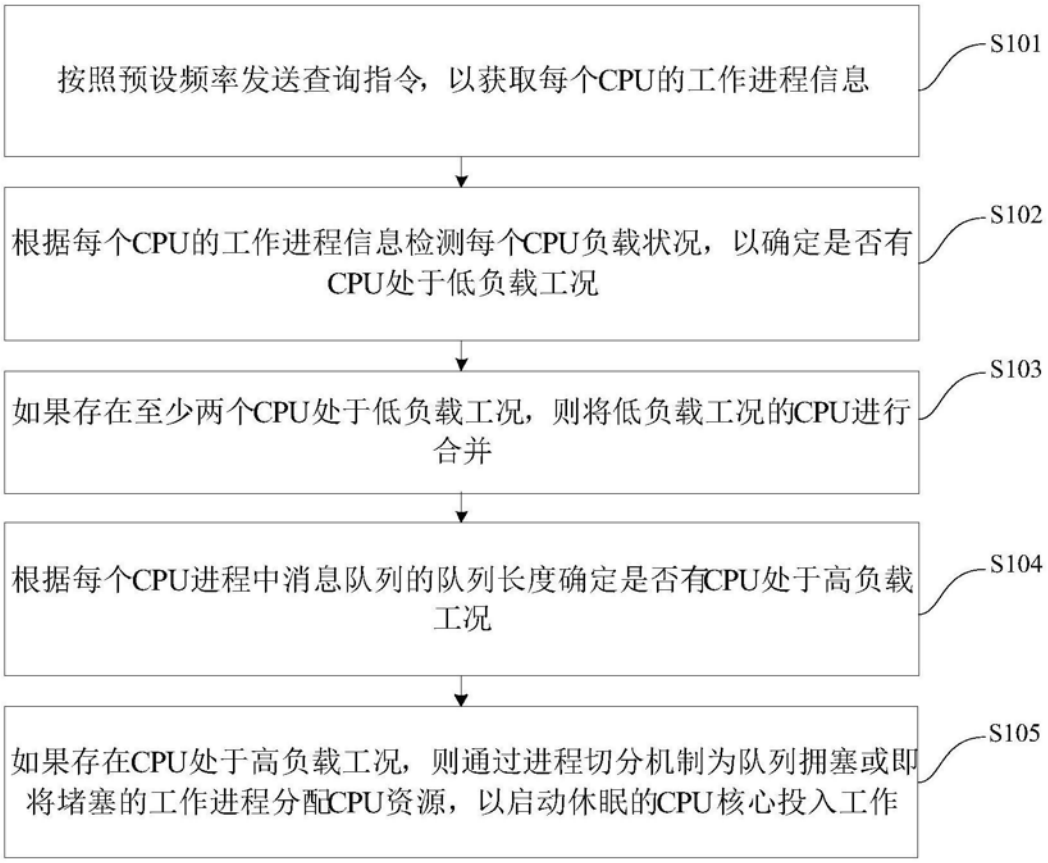

按照预设频率发送查询指令，以获取每个CPU的工作进程信息
S101
根据每个CPU的工作进程信息检测每个CPU负载状况，以确定是否有CPU处于低负载工况
S102
如果存在至少两个CPU处于低负载工况，则将低负载工况的CPU进行合并
S103
根据每个CPU进程中消息队列的队列长度确定是否有CPU处于高负载工况
S104
如果存在CPU处于高负载工况，则通过进程切分机制为队列拥塞或即将堵塞的工作进程分配CPU资源，以启动休眠的CPU核心投入工作
S105

图1

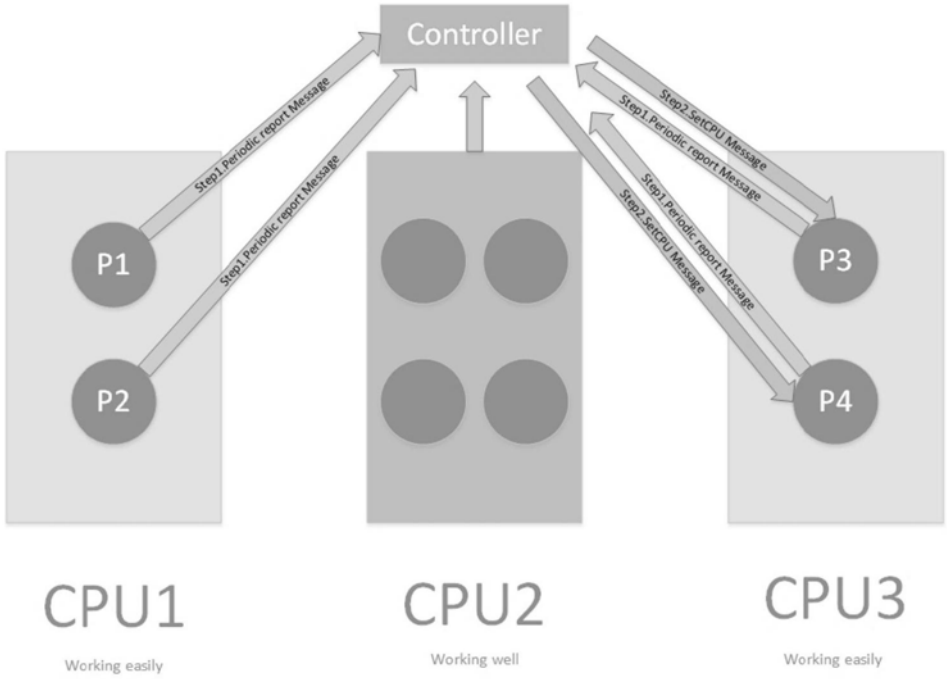

Controller
Step1 Periodic report Message
Step1 Periodic report Message
Step2 SetCPU Message
Step1 Periodic report Message
Step2 SetCPU Message
P1
P2
P3
P4
CPU1
Working easily
CPU2
Working well
CPU3
Working easily

图2

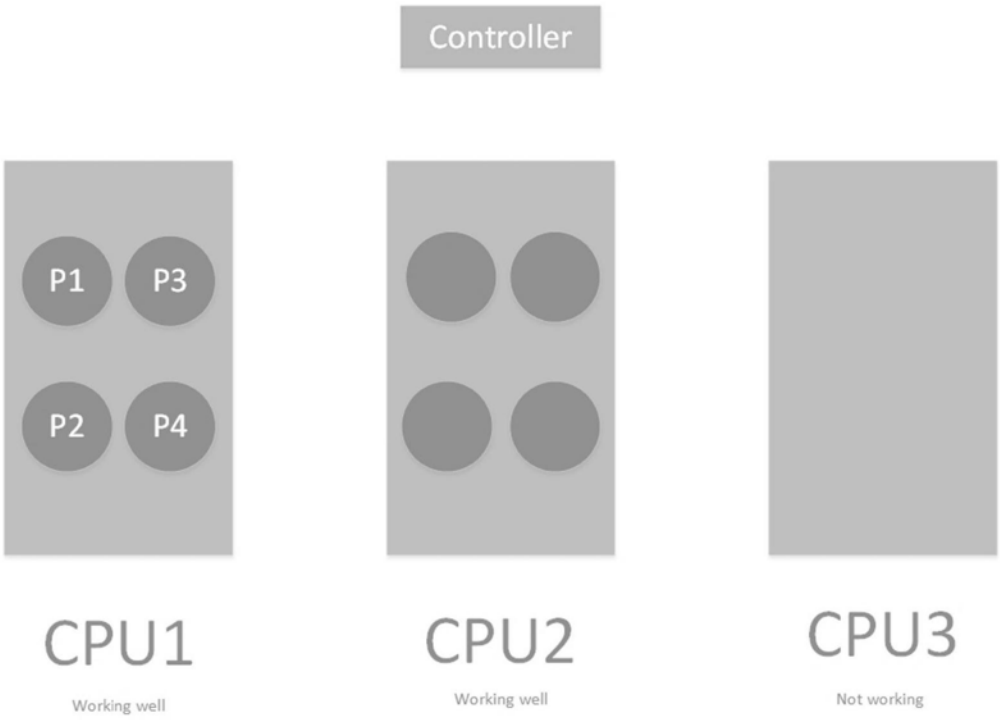

Controller
P1
P3
P2
P4
CPU1
Working well
CPU2
Working well
CPU3
Not working

图3

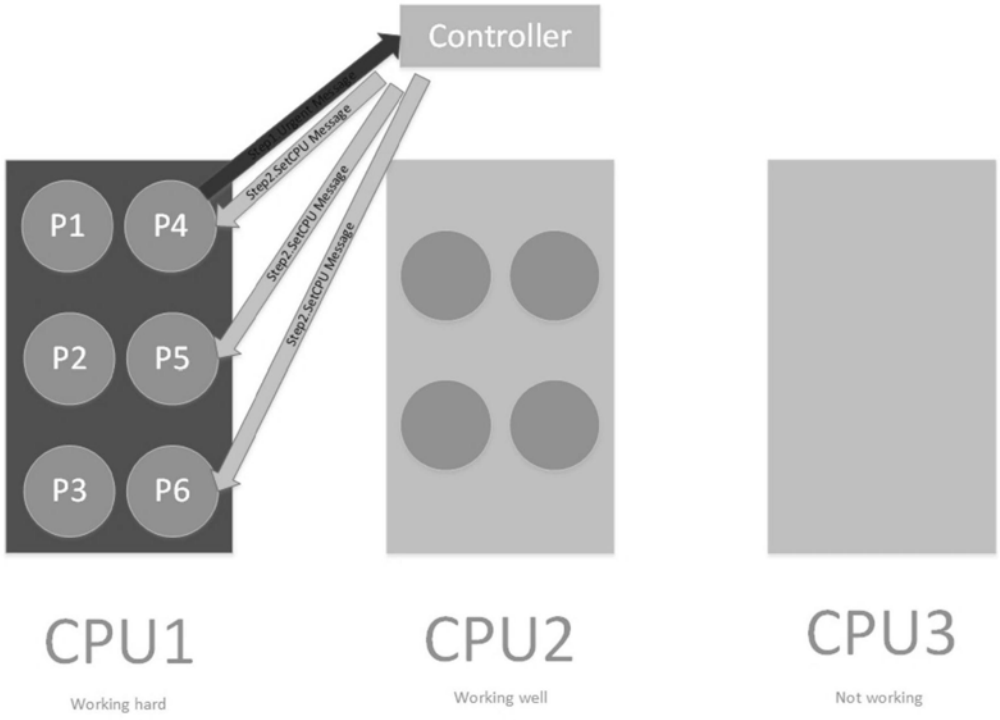

Controller
Step2 SetCPU Message
Step2 SetCPU Message
Step2 SetCPU Message
P1
P4
P2
P5
P3
P6
CPU1
Working hard
CPU2
Working well
CPU3
Not working

图4

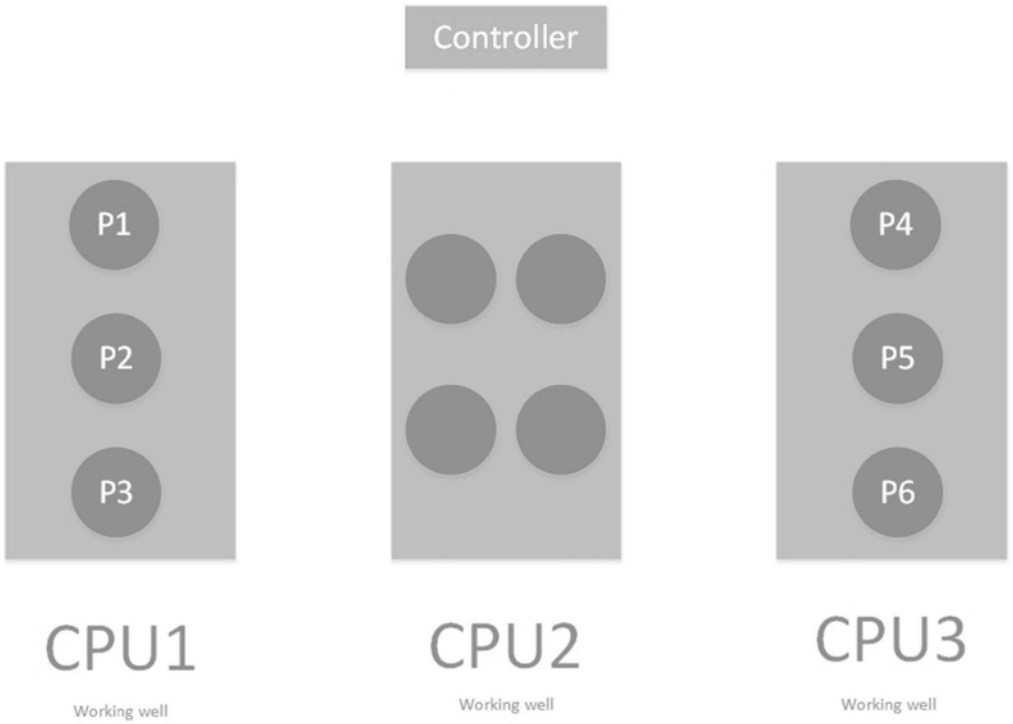

Controller
P1
P2
P3
P4
P5
P6
CPU1
Working well
CPU2
Working well
CPU3
Working well

图5

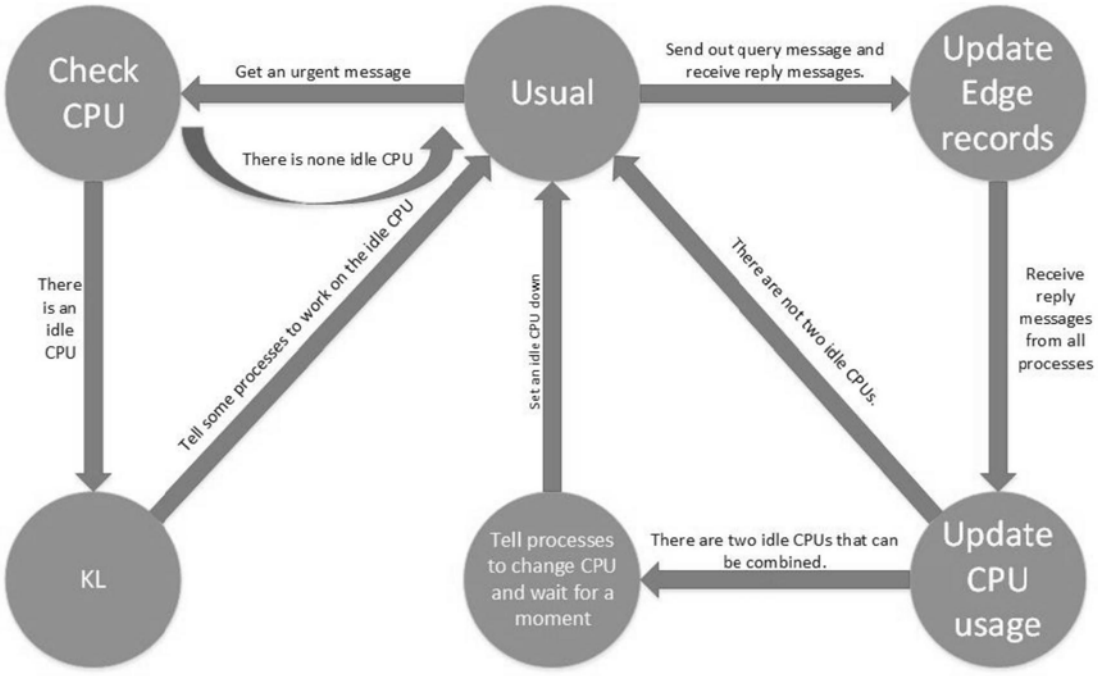

Check CPU
Get an urgent message
Usual
Send out query message and receive reply messages.
Update Edge records
There is none idle CPU
There is an idle CPU
Tell some processes to work on the idle CPU
Set an idle CPU down
There are not two idle CPUs.
Receive reply messages from all processes
KL
Tell processes to change CPU and wait for a moment
There are two idle CPUs that can be combined.
Update CPU usage

图6

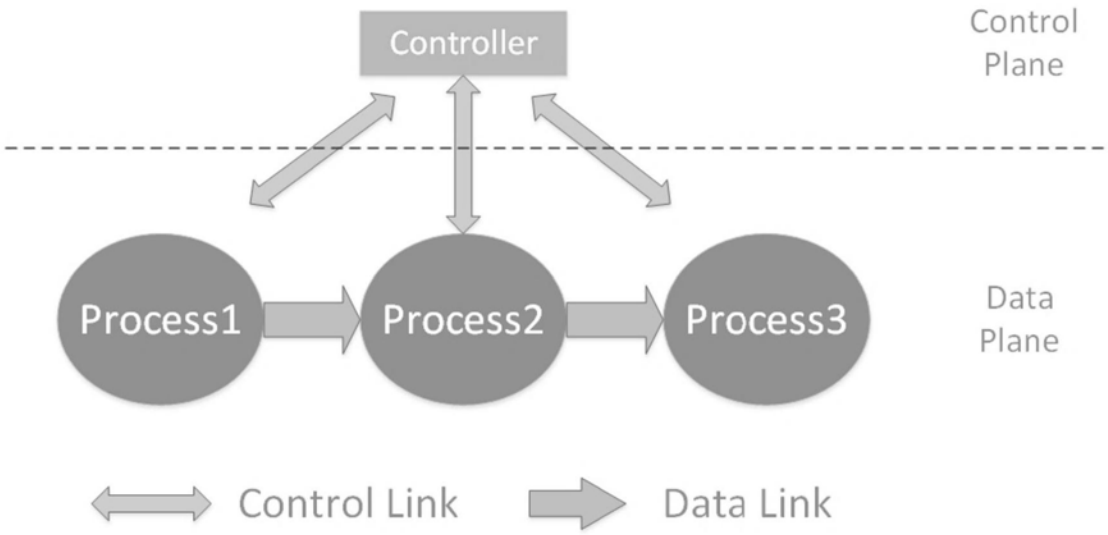

Controller
Control Plane
Process1
Process2
Process3
Data Plane
Control Link
Data Link

图7

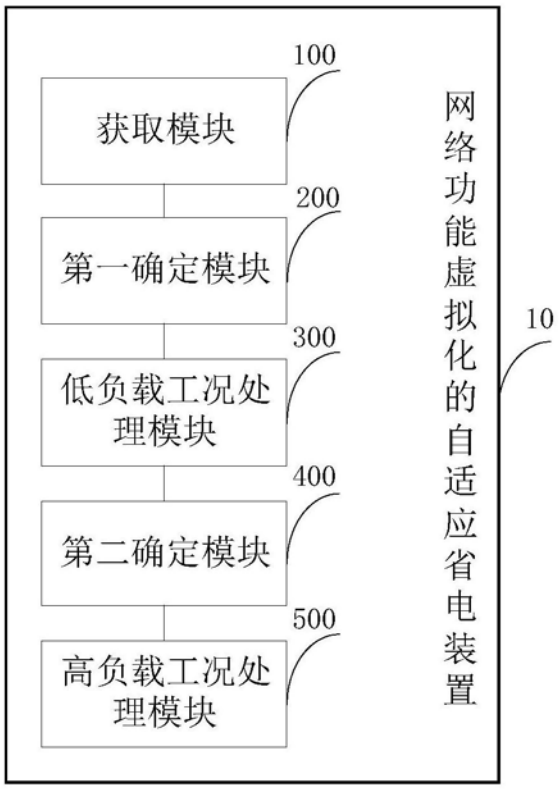

100
获取模块
200
第一确定模块
300
低负载工况处理模块
400
第二确定模块
500
高负载工况处理模块
网络功能虚拟化的自适应省电装置
10

图8