(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 108958944 A
(43)申请公布日 2018.12.07

(21)申请号 201810835572.X

(22)申请日 2018.07.26

(71)申请人 郑州云海信息技术有限公司
地址 450018 河南省郑州市郑东新区心怡路278号16层1601室

(72)发明人 孙昊 赵帅 姜洪正 肖占慧 亓浩

(74)专利代理机构 北京集佳知识产权代理有限公司 11227
代理人 罗满

(51)Int.Cl.
G06F 9/50(2006.01)

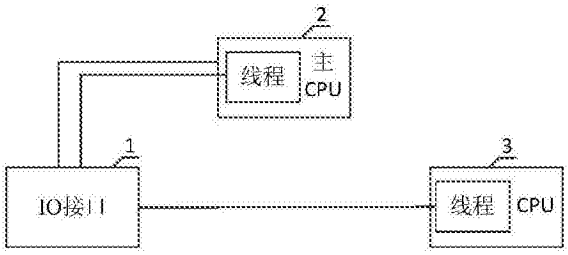
权利要求书1页 说明书6页 附图3页

(54)发明名称

一种多核处理系统及其任务分配方法

(57)摘要

本发明公开了一种多核处理系统，包括多个CPU以及设置于存储设备上的多个IO接口；每个CPU绑定有若干个线程，各个线程分别与各个IO接口一一对应绑定；从多个CPU内选择一个作为主CPU，主CPU用于进行任务调度，分配各个任务所通过的IO接口；IO接口用于将自身接收的任务发送至自身绑定的线程内运行。本发明中同一IO接口的关联性任务能够依据接收顺序由同一个线程依次进行处理，避免了关联性任务分配至不同线程导致的任务整体处理时间长的问题，线程的利用效率和任务处理效率高。本发明还公开了一种基于上述系统的任务分配方法。



CN 108958944 A

1.一种多核处理系统，其特征在于，包括多个CPU以及设置于存储设备上的多个IO接口；

每个所述CPU绑定有若干个线程，各个所述线程分别与各个所述IO接口一一对应绑定；

从多个所述CPU内选择一个作为主CPU，所述主CPU用于进行任务调度，分配各个任务所通过的IO接口；

所述IO接口用于将自身接收的任务发送至自身绑定的线程内运行。

2.根据权利要求1所述的多核处理系统，其特征在于，各个所述CPU绑定的线程数相同。

3.根据权利要求2所述的多核处理系统，其特征在于，每个所述CPU绑定一个线程。

4.根据权利要求1-3任一项所述的多核处理系统，其特征在于，所述主CPU还用于：

检测各个所述线程的利用率，并依据预设调配规则，将利用率超出预设阈值的线程内未运行的部分任务，分配至利用率未超出所述预设阈值的线程内运行。

5.根据权利要求4所述的多核处理系统，其特征在于，所述主CPU还用于：

检测各个任务的处理时间长度，若所述处理时间长度超出预设时间阈值，则按照预设拆分规则将该任务拆分为多个任务。

6.一种多核处理系统的任务分配方法，其特征在于，所述多核处理系统包括多个CPU以及设置于存储设备上的多个IO接口；每个所述CPU绑定有若干个线程；从多个所述CPU内选择一个作为主CPU；所述方法包括：

将各个所述线程分别与各个所述IO接口一一对应绑定；

所述IO接口接收生成的任务后，将自身接收的任务发送至自身绑定的线程内运行。

7.根据权利要求6所述的任务分配方法，其特征在于，各个所述CPU绑定的线程数相同。

8.根据权利要求7所述的任务分配方法，其特征在于，每个所述CPU绑定一个线程。

9.根据权利要求6-8任一项所述的任务分配方法，其特征在于，还包括：

所述主CPU检测各个所述线程的利用率，并依据预设调配规则，将利用率超出预设阈值的线程内未运行的部分，分配至利用率未超出所述预设阈值的线程内运行。

10.根据权利要求9所述的任务分配方法，其特征在于，还包括：

所述主CPU检测各个任务的处理时间长度，若所述处理时间长度超出预设时间阈值，则按照预设拆分规则将该任务拆分为多个任务。

一种多核处理系统及其任务分配方法

技术领域

[0001] 本发明涉及系统任务管理技术领域，特别是涉及一种多核处理系统及其任务分配方法。

背景技术

[0002] 主CPU由于性能需要，使用的都是多CPU(多核)的硬件结构了，Linux操作系统会自动把任务分配到不同的处理器上，并尽可能的保持负载均衡。而负载均衡方式，从操作系统的层面来说，通常是每隔一段时间检查一下每个CPU的负载情况，把任务调整到负载低的核上。

[0003] 由于有些任务属于非独立的任务，而非独立的任务执行时可能需要利用其它任务处理完成后的数据，或者可能需要与其他任务访问相同的IO口资源，由于存储区域在被任务访问时会进行加锁，使其他任务无法访问，因此，这些非独立的任务在执行时需要按顺序进行执行。

[0004] 这种情况下，若依据负载压力对任务进行分配的话，各个任务具体分配到哪个CPU就会变得不确定，可能会将这些非独立任务分配至不同的CPU或者不同的线程进行处理，由于不同线程的处理过程相互独立，因此，可能会使得这些非独立任务的处理顺序发生混乱，导致任务失败。即使任务未失败，某些线程也可能由于自身任务的处理顺序或者访问的内存被锁而需要等待其他线程优先处理，从而导致线程资源浪费，任务整体处理时间长，任务处理效率低。

[0005] 因此，如何提供一种能够提高任务处理效率的多核处理系统及其任务分配方法是本领域技术人员目前需要解决的问题。

发明内容

[0006] 本发明的目的是提供一种多核处理系统及其任务分配方法，关联性任务依据接收顺序由同一个线程依次进行处理，任务整体处理时间短，线程的利用效率和任务处理效率高。

[0007] 为解决上述技术问题，本发明提供了一种多核处理系统，包括多个CPU以及设置于存储设备上的多个IO接口以及主CPU；

[0008] 每个所述CPU绑定有若干个线程，各个所述线程分别与各个所述IO接口一一对应绑定；

[0009] 从多个所述CPU内选择一个作为主CPU，所述主CPU用于进行任务调度，分配各个任务所通过的IO接口；

[0010] 所述IO接口用于将自身接收的任务发送至自身绑定的线程内运行。

[0011] 优选地，各个所述CPU绑定的线程数相同。

[0012] 优选地，每个所述CPU绑定一个线程。

[0013] 优选地，所述主CPU还用于：

[0014] 检测各个所述线程的利用率，并依据预设调配规则，将利用率超出预设阈值的线程内未运行的部分任务，分配至利用率未超出所述预设阈值的线程内运行。

[0015] 优选地，所述主CPU还用于：

[0016] 检测各个任务的处理时间长度，若所述处理时间长度超出预设时间阈值，则按照预设拆分规则将该任务拆分为多个任务。

[0017] 为解决上述技术问题，本发明还提供了一种多核处理系统的任务分配方法，所述多核处理系统包括多个CPU以及设置于存储设备上的多个IO接口；每个所述CPU绑定有若干个线程；从多个所述CPU内选择一个作为主CPU；所述方法包括：

[0018] 将各个所述线程分别与各个所述IO接口一一对应绑定；

[0019] 所述IO接口接收生成的任务后，将自身接收的任务发送至自身绑定的线程内运行。

[0020] 优选地，各个所述CPU绑定的线程数相同。

[0021] 优选地，每个所述CPU绑定一个线程。

[0022] 优选地，还包括：

[0023] 所述主CPU检测各个所述线程的利用率，并依据预设调配规则，将利用率超出预设阈值的线程内未运行的部分，分配至利用率未超出所述预设阈值的线程内运行。

[0024] 优选地，还包括：

[0025] 所述主CPU检测各个任务的处理时间长度，若所述处理时间长度超出预设时间阈值，则按照预设拆分规则将该任务拆分为多个任务。

[0026] 本发明提供了一种多核处理系统及其任务分配方法，令每个线程分别一一对应绑定一个IO接口，这样使得每个IO接口会将任务发送至自身绑定的线程内运行。由于非独立的关联性任务通常会通过同一个IO接口来发送，因此，在本发明中，这些关联性任务会发送至同一个线程内处理，由于一个线程不可能同时处理两个任务，因此，这些关联性任务自然会依据线程的接收顺序依次进行处理，使得各个线程不需要等待其他线程执行完毕后再执行任务，尽可能避免了有些线程任务暂停的情况出现，提高了线程的利用效率和任务处理效率；并且由于这些关联性任务不会同时访问同一存储区域，因此，多数情况下可以省去存储区域加锁的步骤，缩短了任务整体的处理时间。

附图说明

[0027] 为了更清楚地说明本发明实施例中的技术方案，下面将对现有技术和实施例中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本发明的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

[0028] 图1为本发明提供的一种多核处理系统的结构示意图；

[0029] 图2为本发明提供的一种任务分配过程的示意图；

[0030] 图3为一种任务阻塞过程的示意图；

[0031] 图4为本发明提供的另一种任务分配过程的示意图；

[0032] 图5为本发明提供的一种多核处理系统的任务分配方法的过程的流程图。

具体实施方式

[0033] 本发明的核心是提供一种多核处理系统及其任务分配方法，关联性任务依据接收顺序由同一个线程依次进行处理，任务整体处理时间短，线程的利用效率和任务处理效率高。

[0034] 为使本发明实施例的目的、技术方案和优点更加清楚，下面将结合本发明实施例中的附图，对本发明实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例是本发明一部分实施例，而不是全部的实施例。基于本发明中的实施例，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，都属于本发明保护的范围。

[0035] 本发明提供了一种多核处理系统，参见图1和图2所示，图1为本发明提供的一种多核处理系统的结构示意图。图2为本发明提供的一种任务分配过程的示意图。其中，Port为IO接口1。

[0036] 该系统包括多个CPU3以及设置于存储设备上的多个IO接口1；

[0037] 每个CPU3绑定有若干个线程，各个线程分别与各个IO接口1一一对应绑定；

[0038] 从多个CPU3内选择一个作为主CPU2，主CPU2用于进行任务调度，分配各个任务所通过的IO接口1；

[0039] IO接口1用于将自身接收的任务发送至自身绑定的线程内运行。

[0040] 其中，关于任务的定义：任务可以理解为函数，主CPU2处理的任务都是IO相关的内容，所以每个任务包含IO相关的信息，比如IO请求等等。

[0041] 可以理解的是，对于高性能处理器，往往有多个CPU3，即多核处理器，例如可以假设CPU3的核数是6核，即包括6个CPU3。在操作系统层面，基本的运行单位是进程，进程中的调度单元是线程。通常在一个进程中可以包含若干个线程，当然一个进程中至少有一个线程，不然没有存在的意义。线程可以利用进程所拥有的资源，在引入线程的操作系统中，通常都是把进程作为分配资源的基本单位，而把线程作为独立运行和独立调度的基本单位，由于线程比进程更小，基本上不拥有系统资源，故对它的调度所付出的开销就会小得多，能更高效的提高系统多个程序间并发执行的程度。在配置过程中，一个CPU3可以绑定多个线程，即多个线程公用一个CPU3。

[0042] 如图3所示，图3为一种任务阻塞过程的示意图；在现有的系统中，每个线程可能接收多个IO接口1的任务，这样就会导致部分线程任务过多，出现任务阻塞。如图3的情况，线程接收了port2、3、4发送的任务B、C、D，导致出现了任务阻塞。

[0043] 根据存储设备的特点可知，基于IO的任务分为两种，一种任务是独立的，即这种任务可以独立的运行在一个线程上，不依赖其他的任务，可以独立完成；另一种任务是非独立的，需要依赖IO接口1的资源(IO口资源指的是IO口的硬件资源，主要是指转发IO数据包的这条链路，重要指标是IOPS(每秒转发包数)和带宽。)，这些任务之间通常存在关联性，后者可能需要依赖前者的处理结果，或者需要可能需要访问相同的IO口资源，因此这些同一IO接口1发送的一系列的非独立的任务需要按照先后顺序执行，才能避免访问冲突或者任务混乱。

[0044] 根据这个要求，在本发明中，将IO接口1与线程进行了一一绑定，这种连接方式下，同一个IO接口1发送的这类任务会运行在同样的线程上，而不会转移至其他线程处理，尽可

能避免了某些线程需要等待其他线程执行完毕后再执行任务的线程暂停的情况出现，提高了线程的利用效率，并且上述方法能够避免多个线程同时访问相同存储资源时导致的存储区域加锁的情况，因此，多数情况下可以省去存储区域加锁的步骤，缩短了任务整体的处理时间。并且这种设置方式还能够尽可能均衡了每个线程接收的任务数量，减少了任务阻塞的情况出现。如图2所示，当port2的任务B来到，会被分配到线程2，port2的任务B2来到，若B2与B为非独立任务，则B2仍有线程处理。

[0045] 在一种优选实施例中，各个CPU3绑定的线程数相同。

[0046] 可以理解的是，为了提高各个CPU3的利用效率，需要尽可能保证各个CPU3之间的负载均衡，因此，需要各个CPU3执行任务的能力基本保持平衡，为了实现此目的，需要保证各个CPU3绑定相同个数的线程。由于线程与IO口一一对应绑定，因此，线程均衡分布即等同于IO口的均衡分布。可以得知的是，存储设备处理的核心业务是IO流量，每个存储设备都有或多或少的IO口，每个IO口的处理效率跟CPU3资源的分配息息相关。而经过多次实验发现，平均分配IO口资源到CPU3上是比较好的方式，可以获取较高的性能。因此，通过本实施例的设置方式，CPU3的工作效率以及IO口的处理效率较高。

[0047] 进一步可知，每个CPU3绑定一个线程。

[0048] 可以理解的是，经过多次实验发现，每个线程绑定一个CPU3时，系统的性能较高。因此，本实施例中，在主程序启动之初，把每个线程跟CPU3分别绑定，进一步提高了CPU3的处理效率。

[0049] 依据前述内容可知，非独立的任务由于依赖IO资源需要在同一个线程内运行，因此无法将这部分任务分配至其他线程内运行；而独立的任务跟所对应的IO接口1无关，因此，可以运行在任何的线程上，故对于独立的任务，根据线程(CPU3)的忙碌程度，把任务分配到不太忙碌的线程，实现CPU3的均衡配置。

[0050] 作为优选地，主CPU2还用于：

[0051] 检测各个线程的利用率，并依据预设调配规则，将利用率超出预设阈值的线程内未运行的部分任务，分配至利用率未超出预设阈值的线程内运行。

[0052] 以图4为例，图4为本发明提供的另一种任务分配过程的示意图。当port2的任务B来到，会被分配到线程2，当B没有处理完毕，port2的任务C又来了，这时判断出线程2压力较大，同时线程3空闲，所以任务C被分配到线程3。同理，当任务D来到时候，会分配到线程4；当port3的任务E来了，本应该分配到线程3，因为线程3忙，所以会把任务分配到线程5。

[0053] 可以理解的是，预设调配规则中，是根据各个线程的忙碌程度(即利用率)来判断是否需要进行任务调配的，并且，在任务调配时，显然不能将正在处理的任务调配至其他线程，避免任务的中断。另外，依据前述内容可知，本实施例中，调配至其他线程的部分任务均需要为独立任务，即不与其他任务关联、且不依赖IO接口1资源的任务。当然，选择接收任务的线程的方式，本发明不作限定。

[0054] 在另一种优选实施例中，主CPU2还用于：

[0055] 检测各个任务的处理时间长度，若处理时间长度超出预设时间阈值，则按照预设拆分规则将该任务拆分为多个任务。

[0056] 可以理解的是，为了方便任务调度、提高CPU3使用的效率，要求任务的处理时间不能太长。这个太长在CPU3领域里面通常定位为100ms。这个是个估计值，本发明不限定预设

时间阈值的具体数值。

[0057] 另外，当一个任务的可能的处理时间比较长的时候，需要拆分成多个任务。拆分多个任务的方法类似job的机制，本发明对此不作限定。由于这些由同一个任务拆分后得到的子任务之间一般是具有联系的，即这些子任务属于非独立的关联任务，因此，大任务拆分后的子任务优选由同一个线程处理，至于这个线程是接收到大任务的线程还是其他线程(因为有可能接收到大任务的线程过忙，因此将这些拆分后的子任务打包发送至其他线程处理)，本发明不作限定。当然，若拆分后的子任务之间关联性不强的话，也可以将这些子任务分别发送至不同的线程处理，本发明不限定拆分后的任务的处理对象。

[0058] 本发明提供了一种多核处理系统，令每个线程分别一一对应绑定一个IO接口，这样使得每个IO接口会将任务发送至自身绑定的线程内运行。由于非独立的关联性任务通常会通过同一个IO接口来发送，因此，在本发明中，这些关联性任务会发送至同一个线程内处理，由于一个线程不可能同时处理两个任务，因此，这些关联性任务自然会依据线程的接收顺序依次进行处理，使得各个线程不需要等待其他线程执行完毕后再执行任务，尽可能避免了有些线程任务暂停的情况出现，提高了线程的利用效率和任务处理效率；并且由于这些关联性任务不会同时访问同一存储区域，因此，多数情况下可以省去存储区域加锁的步骤，缩短了任务整体的处理时间。

[0059] 本发明还提供了一种多核处理系统的任务分配方法，多核处理系统包括多个CPU以及设置于存储设备上的多个IO接口；每个CPU绑定有若干个线程；从多个CPU内选择一个作为主CPU；参见图5所示，图5为本发明提供的一种多核处理系统的任务分配方法的过程的流程图。

[0060] 该方法包括：

[0061] 步骤s1：将各个线程分别与各个IO接口一一对应绑定；

[0062] 步骤s2：IO接口接收生成的任务后，将自身接收的任务发送至自身绑定的线程内运行。

[0063] 作为优选地，各个CPU绑定的线程数相同。

[0064] 进一步可知，每个CPU绑定一个线程。

[0065] 作为优选地，该方法还包括：

[0066] 主CPU检测各个线程的利用率，并依据预设调配规则，将利用率超出预设阈值的线程内未运行的部分，分配至利用率未超出预设阈值的线程内运行。

[0067] 作为优选地，该方法还包括：

[0068] 主CPU检测各个任务的处理时间长度，若处理时间长度超出预设时间阈值，则按照预设拆分规则将该任务拆分为多个任务。

[0069] 本发明提供了一种多核处理系统的任务分配方法，令每个线程分别一一对应绑定一个IO接口，这样使得每个IO接口会将任务发送至自身绑定的线程内运行。由于非独立的关联性任务通常会通过同一个IO接口来发送，因此，在本发明中，这些关联性任务会发送至同一个线程内处理，由于一个线程不可能同时处理两个任务，因此，这些关联性任务自然会依据线程的接收顺序依次进行处理，使得各个线程不需要等待其他线程执行完毕后再执行任务，尽可能避免了有些线程任务暂停的情况出现，提高了线程的利用效率和任务处理效率；并且由于这些关联性任务不会同时访问同一存储区域，因此，多数情况下可以省去存储

区域加锁的步骤，缩短了任务整体的处理时间。

[0070] 以上的几种具体实施方式仅是本发明的优选实施方式，以上几种具体实施例可以任意组合，组合后得到的实施例也在本发明的保护范围之内。应当指出，对于本技术领域的普通技术人员来说，相关专业技术人员在不脱离本发明精神和构思前提下推演出的其他改进和变化，均应包含在本发明的保护范围之内。

[0071] 还需要说明的是，在本说明书中，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、物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、物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、物品或者设备中还存在另外的相同要素。

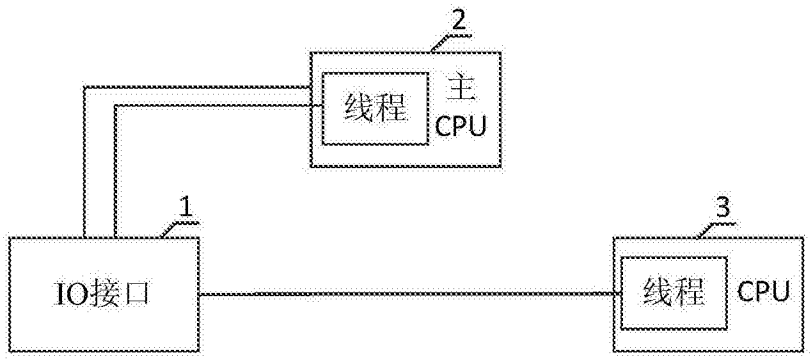

2
线程
主
CPU
1
IO接口
3
线程
CPU

图1

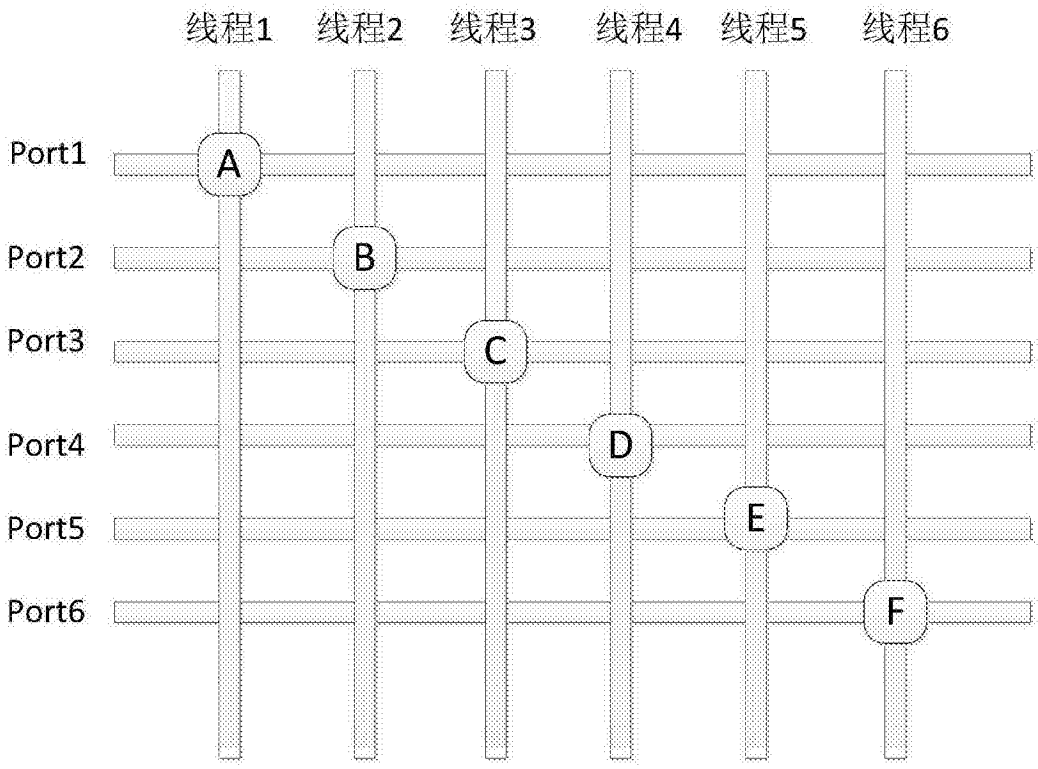

线程1
线程2
线程3
线程4
线程5
线程6
Port1
Port2
Port3
Port4
Port5
Port6
A
B
C
D
E
F

图2

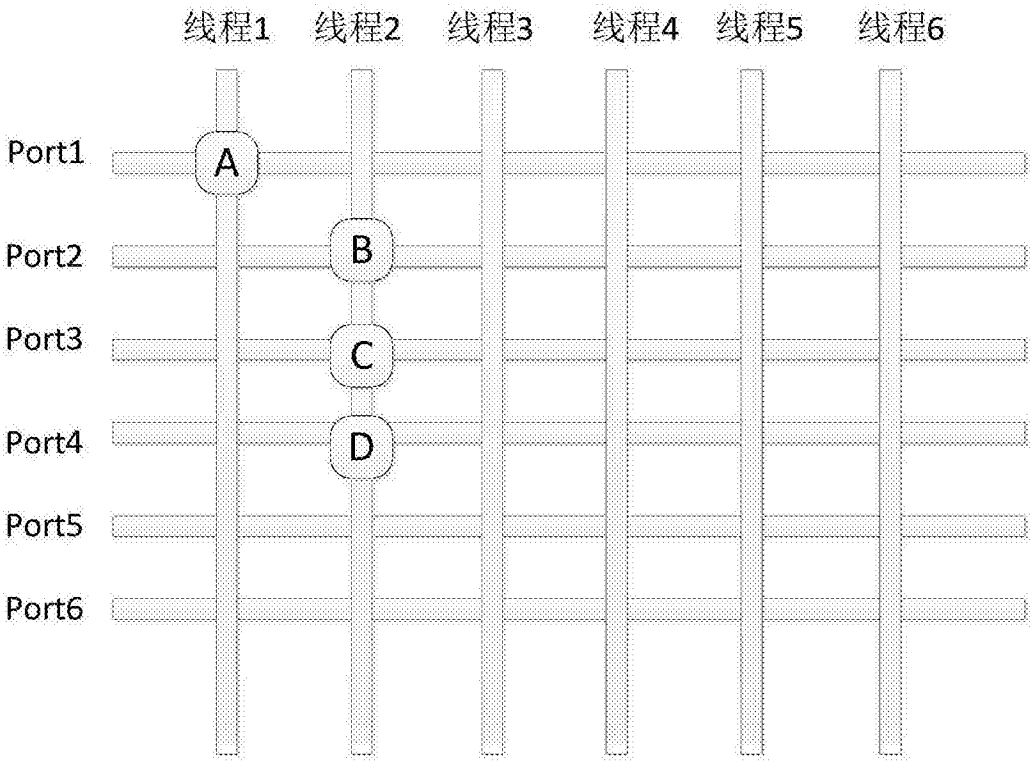

线程1
线程2
线程3
线程4
线程5
线程6
Port1
Port2
Port3
Port4
Port5
Port6
A
B
C
D

图3

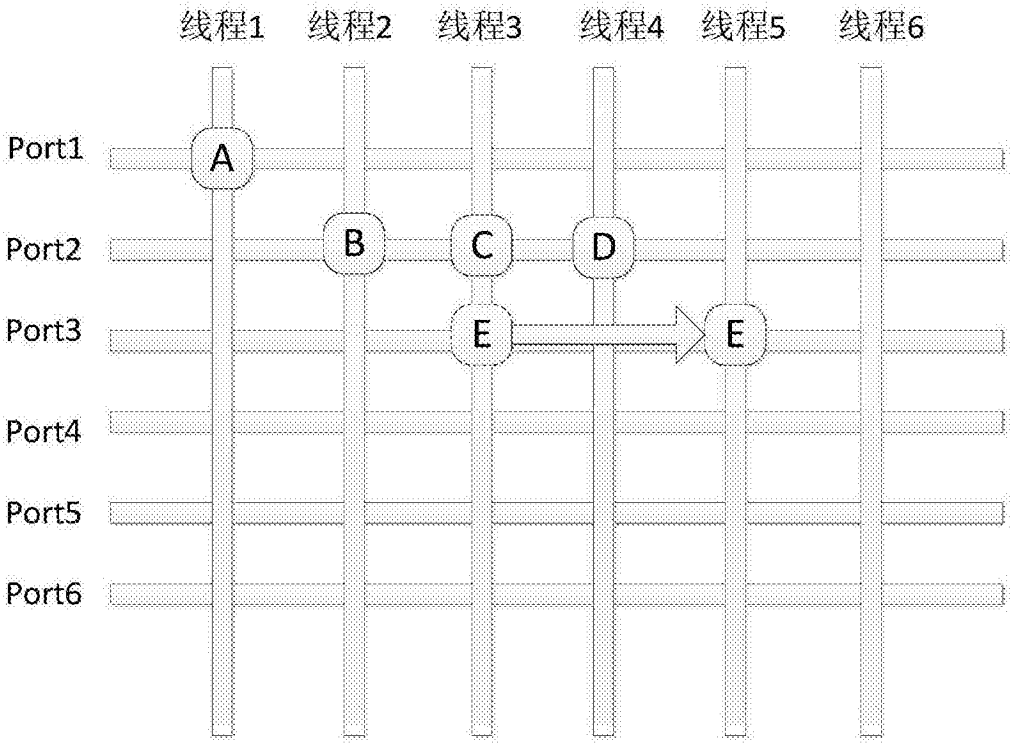

线程1
线程2
线程3
线程4
线程5
线程6
Port1
Port2
Port3
Port4
Port5
Port6
A
B
C
D
E
E

图4

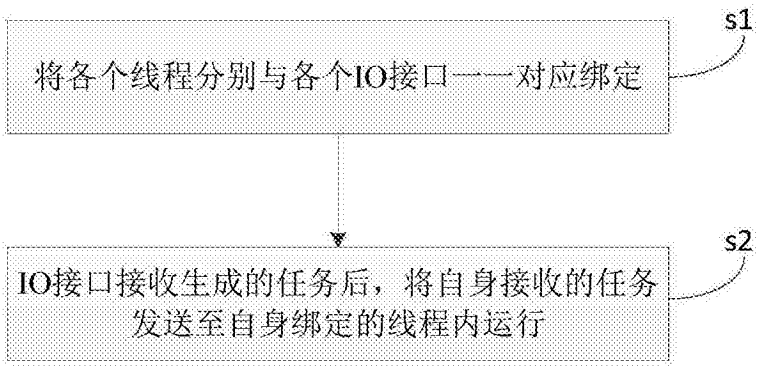

s1
将各个线程分别与各个IO接口一一对应绑定
s2
IO接口接收生成的任务后，将自身接收的任务发送至自身绑定的线程内运行

图5