(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国 际 局

(43) 国际公布日
2024 年 3 月 21 日 (21.03.2024)

WIPO | PCT

(10) 国际公布号
WO 2024/055708 A1

(51) 国际专利分类号:
G06F 9/48 (2006.01)

(21) 国际申请号: PCT/CN2023/104517

(22) 国际申请日: 2023 年 6 月 30 日 (30.06.2023)

(25) 申请语言: 中文

(26) 公布语言: 中文

(30) 优先权:
202211110283.6 2022年9月13日 (13.09.2022) CN

(71) 申请人: 上海寒武纪信息科技有限公司 (**SHANGHAI CAMBRICON INFORMATION TECHNOLOGY CO., LTD.**) [CN/CN]; 中国上海市浦东新区同汇路 168 号 B 座 6 层, Shanghai 201306 (CN)。

(72) 发明人: 冯牧祎 (**FENG, Muyi**); 中国上海市浦东新区同汇路168号B座6层, Shanghai 201306 (CN)。于涌 (**YU, Yong**); 中国上海市浦东新区同汇路168号B座6层, Shanghai 201306 (CN)。韩栋 (**HAN, Dong**); 中国上海市浦东新区同汇路168号B座6层, Shanghai 201306 (CN)。

(74) 代理人: 北京同立钧成知识产权代理有限公司 (**LEADER PATENT & TRADEMARK FIRM**); 中国北京市海淀区西直门北大街32号枫蓝国际A座8F-6, Beijing 100082 (CN)。

(81) 指定国(除另有指明，要求每一种可提供的国家保护): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ,

(54) **Title:** TASK SCHEDULING METHOD AND APPARATUS, AND DEVICE AND MEDIUM

(54) 发明名称: 任务调度方法、装置、设备及介质

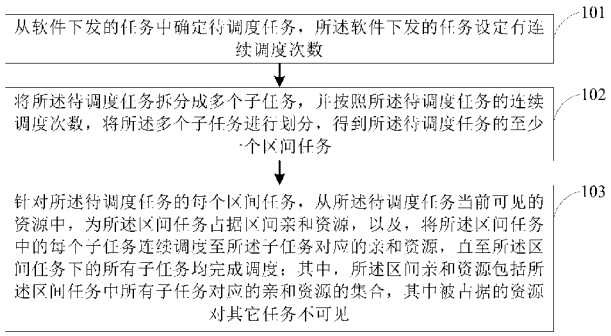

图 2

101 Determine, from among tasks issued by software, a task to be scheduled, wherein the number of instances of consecutive scheduling is set for the tasks issued by the software

102 Split, into a plurality of sub-tasks, the task to be scheduled, and divide the plurality of sub-tasks according to the number of instances of consecutive scheduling of the task to be scheduled, so as to obtain at least one interval task of the task to be scheduled

103 For each interval task of the task to be scheduled, occupy, for the interval task, interval affinity resources from among the current visible resources for the task to be scheduled, and consecutively schedule each sub-task in the interval task to an affinity resource corresponding to the sub-task, until the scheduling of all the sub-tasks in the interval task is completed, wherein the interval affinity resources comprise a set of the affinity resources corresponding to all the sub-tasks in the interval task, and the occupied resources are invisible to the other tasks

(57) **Abstract:** Provided in the present disclosure are a task scheduling method and apparatus, and a device and a medium. The method comprises: after a task to be scheduled is determined, splitting, into a plurality of sub-tasks, the task to be scheduled, and dividing the plurality of sub-tasks according to the number of instances of consecutive scheduling, so as to obtain at least one interval task of the task to be scheduled; and for each interval task of the task to be scheduled, occupying, for the interval task, interval affinity resources from among the current visible resources for the task to be scheduled, and consecutively scheduling each sub-task in the interval task to an affinity resource corresponding to the sub-task, until the scheduling of all the sub-tasks in the interval task is completed. In the scheme, a plurality of sub-tasks are divided according to intervals, and the sub-tasks in the intervals are consecutively scheduled until

WO 2024/055708 A1

[见续页]

LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW。

(84) 指定国(除另有指明，要求每一种可提供的地区保护): ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG)。

根据细则4.17的声明:
— 关于申请人有权要求在先申请的优先权(细则4.17(iii))

本国际公布:
— 包括国际检索报告(条约第21条(3))。

the scheduling of all the sub-tasks in the intervals is completed, such that the locality of data used by all the sub-tasks in the intervals is improved, and the frequency of switching of data in a buffer is reduced, thereby improving the computation efficiency of a chip.

(57) 摘要: 本公开提供一种任务调度方法、装置、设备及介质。其中方法包括确定待调度任务后将待调度任务拆分成多个子任务，并按照连续调度次数，将所述多个子任务进行划分，得到待调度任务的至少一个区间任务；针对待调度任务的每个区间任务，从待调度任务当前可见的资源中，为所述区间任务占据区间亲和资源，以及，将所述区间任务中的每个子任务连续调度至所述子任务对应的亲和资源，直至所述区间任务下的所有子任务均完成调度；本方案通过将多个子任务按区间进行划分，并连续调度所述区间中每个子任务直至所述区间下的所有子任务均完成调度，提升了完成所述区间下所有子任务所使用数据的局部性，减少了数据在缓存器中的切换次数，从而提高芯片的计算效率。

任务调度方法、装置、设备及介质

本公开要求于2022年9月13日提交中国专利局、申请号为202211110283.6、申请名称为“任务调度方法、装置、设备及介质”的中国专利申请的优先权，其全部内容通过引用结合在本公开中。

技术领域

本公开涉及芯片运算技术领域，尤其设计一种任务调度方法、装置、设备及介质。

背景技术

人工智能（Artificial Intelligence，简称AI）芯片做运算时，需要访问大量的运算数据，AI芯片本身的存储数据能力有限，访问随机存取存储器（Random Access Memory，简称RAM）去取得计算相关的数据的延时很高。

目前，AI计算芯片都会通过缓存器来做数据的缓存，在缓存器中缓存当前计算任务所需要的数据，从而减少芯片中计算单元访问RAM需要的延时。

但是，当执行的计算任务相关性不高，需要缓存器缓存不同的数据时，会增加缓存器访问RAM的次数，增加缓存器切换数据的次数，降低了AI芯片的计算效率。

发明内容

本公开提供一种任务调度方法、装置、设备及介质，用来提升芯片计算效率。

一方面，本公开提供一种任务调度方法，包括：

从软件下发的任务中确定待调度任务，所述软件下发的任务设定有连续调度次数；

将所述待调度任务拆分成多个子任务，并按照所述待调度任务的连续调度次数，将所述多个子任务进行划分，得到所述待调度任务的至少一个区间任务；

针对所述待调度任务的每个区间任务，从所述待调度任务当前可见的资源中，为所述区间任务占据区间亲和资源，以及，将所述区间任务中的每个子任务连续调度至所述子任务对应的亲和资源，直至所述区间任务下的所有子任务均完成调度；其中，所述区间亲和资源包括所述区间任务中所有子任务对应的亲和资源的集合，其中被占据的资源对其它任务不可见。

在一种实施例中，所述方法还包括：

若所述区间任务中的所有子任务均完成调度，则释放所述区间任务占据的所述区间亲和资源，其中被释放的资源对其它任务可见。

在一种实施例中，从所述待调度任务当前可见的资源中，为所述区间任务占据区间亲和资源，包括：

获取所述区间任务中子任务对应的资源掩码，所述子任务对应的资源掩码表征所述子任务对应的亲和资源；

将所述待调度任务当前可见的资源中所述掩码表征的资源，作为所述区间亲和资源，为所述区间任务占用所述区间亲和资源。

在一种实施例中，所述方法还包括：

通过解析所述任务调度指令，获得所述连续调度次数并记录在第一寄存器中；

基于第一更新条件，更新第二寄存器中的数据，所述第二寄存器中的数据表征当前已连续调度的次数；其中，所述第一更新条件包括：若当前已成功连续调度的次数达到所述第一寄存器中的连续调度次数，或者执行所述将所述待调度任务拆分成多个子任务的步骤后，或者收到针对所述待调度任务的终止调度指令，或者当前达到最大并行度，则对所述第二寄存器清零；否则，每次完成子任务的调度后，将所述第二寄存器执行加 1 处理；

基于第二更新条件，更新第三寄存器中的数据，所述第三寄存器中的数据表征当前需为所述区间任务占据的区间亲和资源；其中，所述第二更新条件包括：若当前已成功连续调度的次数达到所述第一寄存器中的连续调度次数，或者执行所述将所述待调度任务拆分成多个子任务的步骤后，或者收到针对所述待调度任务的终止调度指令，或者当前达到最大并行度，则清除所述第三寄存器中的数据；否则，每次完成子任务的调度后，从所述第三寄存器中删除该子任务的亲和资源。

在一种实施例中，所述将所述区间任务中的每个子任务连续调度至所述子任务对应的亲和资源，包括：

若第二寄存器中的数据未达到第一寄存器中的连续调度次数，则确定所述区间任务中未完成调度的子任务；

从所述未完成调度的子任务中选取待调度的子任务，将所述待调度的子任务调度至所述子任务对应的亲和资源，直至所述区间任务中所有子任务均完成调度。

在一种实施例中，所述软件下发的任务具有优先级；所述从软件下发的任务中确定待调度任务，包括：

根据当前的可用资源，按照优先级自高向低，从每个优先级对应的任务中选取至少一个任务作为所述待调度任务，直至当前的可用资源不包括任意任务对应的亲和资源；其中，不同待调度任务对应的亲和资源不同，所述可用资源包括除当前的所有待调度任务对应的亲和资源以外的资源。

在一种实施例中，所述从每个优先级对应的任务中选取至少一个任务作为所述待调度任务，包括：

若所述优先级对应的任务中存在亲和资源有重合的不同任务，则将当前所述不同任务中权重最大的任务作为所述待调度任务，并增加所述不同任务中其它任务的权重。

另一方面，本公开提供一种任务调度装置，包括：

确定模块，用于从软件下发的任务中确定待调度任务，所述软件下发的任务设定有连续调度次数；

处理模块，用于将所述待调度任务拆分成多个子任务，并按照所述待调度任务的连续调度次数，将所述多个子任务进行划分，得到所述待调度任务的至少一个区间任务；

处理模块，还用于针对所述待调度任务的每个区间任务，从所述待调度任务当前可见的资源中，为所述区间任务占据区间亲和资源，以及，将所述区间任务中的每个子任务连续调度至所述子任务对应的亲和资源，直至所述区间任务下的所有子任务均完成调度；其中，所述区间亲和资源包括所述区间任务中所有子任务对应的亲和资源的集合，其中被占据的资源对其它任务不可见。

在一种实施例中，所述处理模块，还用于若所述区间任务中的所有子任务均完成调度，则释放所述区间任务占据的所述区间亲和资源，其中被释放的资源对其它任务可见。

在一种实施例中，所述处理模块，具体用于获取所述区间任务中子任务对应的资源掩码，所述子任务对应的资源掩码表征所述子任务对应的亲和资源；

所述处理模块，具体还用于将所述待调度任务当前可见的资源中所述掩码表征的资源，作为所述区间亲和资源，为所述区间任务占用所述区间亲和资源。

在一种实施例中，所述处理模块，具体还用于通过解析所述任务调度指令，获得所述连续调度次数并记录在第一寄存器中；

所述处理模块，具体还用于基于第一更新条件，更新第二寄存器中的数据，所述第二寄存器中的数据表征当前已连续调度的次数；其中，所述第一更新条件包括：若当前已成功连续调度的次数达到所述第一寄存器中的连续调度次数，或者执行所述将所述待调度任务拆分成多个子任务的步骤后，或者收到针对所述待调度任务的终止调度指令，或者当前达到最大并行度，则对所述第二寄存器清零；否则，每次完成子任务的调度后，将所述第二寄存器执行加 1 处理；

所述处理模块，具体还用于基于第二更新条件，更新第三寄存器中的数据，所述第三寄存器中的数据表征当前需为所述区间任务占据的区间亲和资源；其中，所述第二更新条件包括：若当前已成功连续调度的次数达到所述第一寄存器中的连续调度次数，或者执行所述将所述待调度任务拆分成多个子任务的步骤后，或者收到针对所述待调度任务的终止调度指令，或者当前达到最大并行度，则清除所述第三寄存器中的数据；否则，每次完成子任务的调度后，从所述第三寄存器中删除该子任务的亲和资源。

在一种实施例中，所述处理模块，具体还用于若第二寄存器中的数据未达到第一寄存器中的连续调度次数，则确定所述区间任务中未完成调度的子任务；

所述处理模块，具体还用于从所述未完成调度的子任务中选取待调度的子任务，将所述待调度的子任务调度至所述子任务对应的亲和资源，直至所述区间任务中所有子任务均完成调度。

在一种实施例中，所述处理模块，具体还用于根据当前的可用资源，按照优先级自高向低，从每个优先级对应的任务中选取至少一个任务作为所述待调度任务，直至当前的可用资源不包括任意任务对应的亲和资源；其中，不同待调度任务对应的亲和资源不同，所述可用资源包括除当前的所有待调度任务对应的亲和资源以外的资源。

在一种实施例中，所述处理模块，具体还用于若所述优先级对应的任务中存在亲和资源有重合的不同任务，则将当前所述不同任务中权重最大的任务作为所述待调度任务，并增加所述不同任务中其它任务的权重。

又一方面，本公开提供一种电子设备，包括：处理器，以及与所述处理器通信连接的存储器；

所述存储器存储计算机执行指令；

所述处理器执行所述存储器存储的计算机执行指令，以实现如前所述的方法。

又一方面，本公开提供一种计算机可读存储介质，所述计算机可读存储介质中存储有计算机执行指令，所述计算机执行指令被处理器执行时用于实现如前所述的方法。

本公开提供的任务调度方法、装置、设备及介质中，根据软件下发的任务确定出待调度任务，并将所述待调度任务拆分成多个子任务，以及根据解析出的所述待调度任务对应的连续调度次数，将所述多个子任务按区间进行划分，针对所述待调度任务的每个区间任务，从所述待调度任务当前可见的资源中，为所述区间任务占据区间亲和资源，将所述区间任务中的每个子任务连续调度至所述子任务对应的亲和资源，直至所述区间任务下的所有子任务均完成调度。本方案通过将多个子任务按区间进行划分，并连续调度所述区间中每个子任务直至所述区间下的所有子任务均完成调度，由于执行同一任务下的子任务需要的数据具有强相关性，所以连续调度同一区间下的子任务可以有效减少缓存器对于 RAM 的访存，减少了数据在缓存器中的切换次数，从而有效提升芯片计算效率。

附图说明

此处的附图被并入说明书中并构成本说明书的一部分，示出了符合本公开的实施例，并与说明书一起用于解释本公开实施例的原理。

图 1 为示例的运算数据访问过程；

图 2 为本公开实施例一提供的任务调度方法的流程示意图；

图 3 为本公开实施例一提供的另一种任务调度方法的流程示意图；

图 4 为本公开实施例一提供的又一种任务调度方法的流程示意图；

图 5 为本公开实施例三提供的任务调度装置的结构示意图；

图 6 是根据一示例性实施例示出的一种中央控制单元的装置框图；

图 7 为本公开实施例五中提供的一种电子设备的结构示意图。

通过上述附图，已示出本公开明确的实施例，后文中将有更详细的描述。这些附图和文字描述并不是为了通过任何方式限制本公开构思的范围，而是通过参考特定实施例为本领域技术人员说明本公开的概念。

具体实施方式

这里将详细地对示例性实施例进行说明，其示例表示在附图中。下面的描述涉及附图时，除非另有表示，不同附图中的相同数字表示相同或相似的要素。以下示例性实施例中所描述的实施方式并不代表与本公开相一致的所有实施方式。相反，它们仅是与如所附权利要求书中所详述的、本公开的一些方面相一致的装置和方法的例子。

需要说明的是，本公开中对于术语的简要说明，仅是为了方便理解接下来描述的实施方式，而不是意图限定本公开的实施方式。除非另有说明，这些术语应当按照其普通和通常的含义理解。

图1为示例的运算数据访问过程，芯片本身的存储数据非常有限，大部分运算数据均存储在RAM中，然而访问RAM的延时很高。当芯片做运算时，芯片完成一个任务需要用到大量的计算单元，所述计算单元都需要相应的运算数据来完成相关的运算，若由芯片中的计算单元直接从RAM中调取资源，芯片的计算效率会很低。所以通过增设缓存器，缓存器提前从RAM中调取当前芯片所执行的任务所需要的运算数据，再将所述运算资源缓存在缓存器中，由芯片上的计算单元直接从缓存器中调取需要的运算数据，减少芯片上的计算单元直接从RAM中调取数据的延时时间，提高芯片的计算效率。

下面以具体的实施例对本公开的技术方案以及本公开的技术方案进行详细说明。下面这几个具体的实施例可以相互结合，对于相同或相似的概念或过程可能在某些实施例中不再赘述。在本公开的描述中，除非另有明确的规定和限定，各术语应在本领域内做广义理解。下面将结合附图，对本公开的实施例进行描述。

实施例一

图2为本公开实施例一提供的任务调度方法的流程示意图，如图2所示，该方法包括：

步骤101、从软件下发的任务中确定待调度任务，所述软件下发的任务设定有连续调度次数；

步骤102、将所述待调度任务拆分成多个子任务，并按照所述待调度任务的连续调度次数，将所述多个子任务进行划分，得到所述待调度任务的至少一个区间任务；

步骤103、针对所述待调度任务的每个区间任务，从所述待调度任务当前可见的资源中，为所述区间任务占据区间亲和资源，以及，将所述区间任务中的每个子任务连续调度至所述子任务对应的亲和资源，直至所述区间任务下的所有子任务均完成调度；其中，所述区间亲和资源包括所述区间任务中所有子任务对应的亲和资源的集合，其中被占据的资源对其它任务不可见。

结合场景示例：在软件运行的过程中，会生成大量需要被执行的任务，软件会将所述任务下发给芯片执行。软件下发的任务被称为Kernel任务，所述Kernel任务的粒度比较大，可以拆分成多个子任务。当软件给芯片以Kernel任务粒度的大小给芯片下发任务时，会同时下发所述Kernel任务对应的连续调度属性，即该Kernel任务可以连续调度的次数，作为一个参数写在kernel里，所述连续调度属性可通过软件灵活性设置。芯片在收到任务后，会在大量需要被执行的kernel任务中确认出当前时刻下需要被执行的任务，也就是当前的待调度任务。并通过解析当前的待调度任务的连续调度参数控制连续调度的区间，

从而让软件灵活的使用连续调度属性，所述连续调度指的是不间断的调度同一个连读调度区间中的子任务，直到一个连读调度区间中的子任务全部被调度完成。比如，若当前的待调度任务是一个可以被拆分成 10 个子任务的 kernel 任务，所述 10 个子任务分别为 job0-job9，并且软件指定的连读调度属性为 4，就以 4 个子任务为一个区间进行划分，比如 job0-job3 为一个区间，job4-job7 为一个区间，再将剩下的 job8 与 job9 作为一个区间，同一个区间中的子任务进行不间断的连续调度。

芯片完成每个调度任务除了需要相关的运算数据外，还需要占据一定的计算资源，所述计算资源可以理解为计算单元，芯片需要通过调度任务对应的运算数据在一定的计算单元内完成每个调度任务。在一个芯片中，一共有 16 个 IPU cluster，然后每个 IPU cluster 包含 4 个 IPU core，所述一个 IPU core 就是一个计算单元。每个 kernel 任务拆分出多个子任务，其中所述子任务具有两种类型，可以分为 block 任务与 UnionX 任务，由于 block 任务与 UnionX 任务的任务类型不同，所以需要占据的计算单元的多少不同，其中完成一个 block 任务需占用一个 IPU core，完成一个 UnionX 任务需要占用 X 个 IPU cluster，同一个 kernel 任务下的所有子任务的任务类型一致。

芯片中的所有计算单元并不全都符合完成某一个任务的条件，软件下发的 kernel 任务以及拆分出的子任务都有一些特定对应的一些计算单元来完成，可以完成某一任务的所有计算单元都是该任务对应的亲和资源。每个任务除了需要对应的运算数据外，还需要计算单元才可完成，但是所述需要的计算单元必须是任务对应的亲和资源。

为了完成一个区间中所有子任务的连续调度，当一个区间中的首个子任务被调度成功后，那么所述区间中每个子任务对应的所有亲和资源对其他 kernel 任务不可见，其他 kernel 任务的子任务不可以再占用已不可见的计算单元，所述区间中每个子任务对应的所有亲和资源只为所述区间中的子任务提供计算单元。同一个 kernel 任务下的所有子任务对应的亲和资源是一致的，比如，当上述 job0-job3 这一个区间中的首个任务，也就是 job0 被调度成功后，假如 job0 所属的 kernel 下的 job 任务需要的亲和资源有 IPU cluster1、IPU cluster2 与 IPU cluster3，那当 job0 被调度成功后，将占据 IPU cluster1、IPU cluster2 与 IPU cluster3，并对其他 kernel 任务不可见。往往一个任务对应的亲和资源都比完成该任务需要的实际资源要大很多，比如完成 job0 只需要用到 IPU cluster1 中的一个计算单元，就是说完成 job0 只需要占用 IPU cluster1 中的一个 IPU core，那么在 IPU cluster1 中剩余的其他 3 个 IPU core 将作为同一区间中其他子任务可供选择的亲和资源。

本示例通过将多个子任务按区间进行划分，并通过占据所述区间中所有子任务的亲和资源对其他 kernel 任务不可见，连续调度所述区间中每个子任务，直至所述区间下的所有子任务均完成调度。因为来自同一个 kernel 任务的处理数据会有强相关性，比如指令地址、数据地址、参数等信息会有大量的重复，所以当计算单元执行来自同一个 kernel 任务的子任务时，可以有效提升访问 cache 的命中率，从而有效提升芯片计算效率。

在一种示例中，若所述区间任务中的所有子任务均完成调度，则释放所述区间任务占据的所述区间

亲和资源，其中被释放的资源对其它任务可见。

当该区间中每个子任务均被调度成功到亲和资源上并完成每个子任务的计算后，将释放被占据的资源对其他 kernel 任务可见。比如当 job0-job3 均被成调度功到对应的亲和资源上，并在计算资源上完成job0-job3的计算后，会释放所述job0-job3的亲和资源IPU cluster1、IPU cluster2与IPU cluster3对其他kernel任务可见。

在一种示例中，图 3 为本公开实施例一提供的另一种任务调度方法的流程示意图，在步骤 101 中，从所述待调度任务当前可见的资源中，为所述区间任务占据区间亲和资源，包括：

步骤 201、获取所述区间任务中子任务对应的资源掩码，所述子任务对应的资源掩码表征所述子任务对应的亲和资源；

步骤 202、将所述待调度任务当前可见的资源中所述掩码表征的资源，作为所述区间亲和资源，为所述区间任务占用所述区间亲和资源。

在一个芯片上一共有 16 个 IPU cluster，每个 Kernel 任务都有各自对应的亲和资源，每个 Kernel 任务对应的亲和资源中包括的 IPU cluster 可用资源掩码来对应，所述资源掩码可以采用二进制数字来表征。比如针对 job0 所属的 Kernel 任务来说，每个 IPU cluster 若是 job0 的亲和资源可用 1 来表示，若不是 job0 的亲和资源可用 0 来表示。假如 job0 的亲和资源有 IPU cluster1、IPU cluster2 与 IPU cluster3，那么只有 IPU cluster1、IPU cluster2 与 IPU cluster3 为 1，其他 IPU cluster 均为 0，那么用来表征 job0 的亲和资源的资源掩码为 0000000000001110，资源掩码的数位从右到左依次代表 IPU cluster0-IPU cluster15 对 Kernel 任务亲和资源的表征结果。本实例采用资源掩码的方式表征子任务对应的亲和资源，可在调度子任务时准确调度到每个子任务对应的亲和资源上。

在一种示例中，图 4 为本公开实施例一提供的又一种任务调度方法的流程示意图，包括：

步骤 301、通过解析所述任务调度指令，获得所述连续调度次数并记录在第一寄存器中；

步骤 302、基于第一更新条件，更新第二寄存器中的数据，所述第二寄存器中的数据表征当前已连续调度的次数；其中，所述第一更新条件包括：若当前已成功连续调度的次数达到所述第一寄存器中的连续调度次数，或者执行所述将所述待调度任务拆分成多个子任务的步骤后，或者收到针对所述待调度任务的终止调度指令，或者当前达到最大并行度，则对所述第二寄存器清零；否则，每次完成子任务的调度后，将所述第二寄存器执行加 1 处理；

步骤 303、基于第二更新条件，更新第三寄存器中的数据，所述第三寄存器中的数据表征当前需为所述区间任务占据的区间亲和资源；其中，所述第二更新条件包括：若当前已成功连续调度的次数达到所述第一寄存器中的连续调度次数，或者执行所述将所述待调度任务拆分成多个子任务的步骤后，或者收到针对所述待调度任务的终止调度指令，或者当前达到最大并行度，则清除所述第三寄存器中的数据；否则，每次完成子任务的调度后，从所述第三寄存器中删除该子任务的亲和资源。

根据场景示例，执行连续调度的过程中，当满足连续调度停止的条件时，连续调度将停止，在此之

前，需要首先解析出软件下发的 kernel 任务的连续调度参数，当解析出所述参数后，将其记录在芯片中的第一寄存器中，以供后续参考。除了第一寄存器，芯片中还存在第二寄存器与第三寄存器，所述第二寄存器主要的任务是维护一个表示当前连续调度次数的表格，在连续调度的过程中，基于第一更新条件更新所述表格，所述第一更新条件除了包括次数清零条件，好包括若子任务每成功调度一次，则将表格中的次数加 1。所述清零条件包括：条件 1：当前已成功连续调度的次数达到所述第一寄存器中的连续调度次数；条件 2：执行所述将待调度任务被拆分成多个子任务的步骤后；条件 3：收到针对所述待调度任务的终止调度指令；条件 4：当前达到最大并行度。

所述条件 1 指的是连续调度的次数达到所述第一寄存器中记录的连续调度参数，当前连续调度区间中的所有子任务均被调度成功，所以对所述第二寄存器中维护的表格清零以方便记录下次连续调度的次数。所述条件 2 指的是待调度任务刚被拆分成多个子任务需要进行连续调度，相当于初始化清零。所述条件 3 指的是当前连续调度的过程中，存在子任务不惜要被执行，则清楚目前连续调度的次数以方便记录下次连续调度的次数。所述条件 4，最大并行度是指一个 kernel 最多占用的资源数量，若一个 kernel 只允许被占用 4 个 cluster，那 4 就是他的最大并行度，当目前同时执行的子任务达到对的 kernel 的最大并行度，则清楚目前连续调度的次数以方便记录下次连续调度的次数。

所述第三寄存器主要的任务是记录当前待调度任务可占用的资源，并生成一个动态表格，在连续调度的过程中，基于第二更新条件更新所述动态表格，所述第二更新条件除了包括清零条件外，当连读调度区间内每个子任务被调度成功后，从所述第三寄存器中删除该子任务的亲和资源。具体的，比如 job0-job3 这一个区间中子任务对应的亲和资源为 IPU cluster1、IPU cluster2 与 IPU cluster3，当 job0 调度成功后，此时所述第三寄存器中动态表格将删除所述 IPU cluster1、IPU cluster2 与 IPU cluster3，所述 IPU cluster1、IPU cluster2 与 IPU cluster3 将对其他待调度任务不可见。

需要说明的是，若执行 job0 需要占用 IPU cluster1 中的一个 IPU core，当 job0 调度成功后，与所述 job0 同一个区间内的其他子任务不会再占用所述 job0 占用的 IPU core，但是当 job0 被执行完毕，不需要再占用计算资源时，可对同一个区间内的其他子任务释放出所占用的 IPU core。

在一种示例中，若是在此区间内的所有子任务在全部调度成功之前，之前已被调度成功的子任务已完成，则执行该子任务所占用的亲和资源会被闲置出来，所以在动态表格下一次更新后，会显示出被闲置的亲和资源。比如在 job3 调度成功后，job0 已完成，那被 job0 占用的 IPU cluster1 中的一个 IPU core 会显示在此时的动态表格中。

本示例通过维护表示当前连续调度次数的表格与当前子任务可占用的资源的动态表格，直观的表现出当前连续调度的次数与资源占用情况，避免造成资源的浪费与冲突。

在一种示例中，所述将所述区间任务中的每个子任务连续调度至所述子任务对应的亲和资源，包括：

若第二寄存器中的数据未达到第一寄存器中的连续调度次数，则确定所述区间任务中未完成调度的子任务；

从所述未完成调度的子任务中选取待调度的子任务，将所述待调度的子任务调度至所述子任务对应的亲和资源，直至所述区间任务中所有子任务均完成调度。

当连续调度的任务在正常执行时，所述第二寄存器中维护的表格是表示当前连续调度次数的表格，当表格中记录的次数没有达到所述第一寄存器中记录的参数时，说明当权连续调度区间中仍有没有被调度，则从没有被调度的子任务中选取待调度子任务，将当前的待调度子任务调度到对应的亲和资源上，直到当前连续调度区间中的子任务都被调度完成。本示例通过参考表示当前连续调度次数的表格与当前子任务可占用的资源的动态表格，准确的选取待调度子任务与当前可调度的亲和资源，避免造成资源的浪费与冲突。

本实施例过将多个子任务按区间进行划分，并通过占据所述区间中所有子任务的亲和资源对其他 kernel 任务不可见，连续调度所述区间中每个子任务，直至所述区间下的所有子任务均完成调度。并通过参考表示当前连续调度次数的表格与当前子任务可占用的资源的动态表格，为子任务的调度准确的选取亲和资源，因为连续调度区间中的所有子任务都时来自同一个 kernel 任务，子任务之间的数据具有强相关性，所以可以有效提升访问 cache 的命中率，减少了数据在缓存器中的切换次数，从而有效提升芯片计算效率。

实施例 2

在连读调度的过程中，首先把握 kernel 任务的优先级，因为 kernel 任务分为不同的优先等级，高等级的 kernel 任务优先于低等级的 kernel 任务占用亲和资源，并且在同等级的 kernel 任务中，粒度大的 kernel 任务优先于粒度小的 kernel 任务占用亲和资源。

在一种示例中，所述软件下发的任务具有优先级；所述从软件下发的任务中确定待调度任务，包括：

根据当前的可用资源，按照优先级自高向低，从每个优先级对应的任务中选取至少一个任务作为所述待调度任务，直至当前的可用资源不包括任意任务对应的亲和资源；其中，不同待调度任务对应的亲和资源不同，所述可用资源包括除当前的所有待调度任务对应的亲和资源以外的资源。

结合场景示例，当同时存在多个 kernel 任务需要被调度时，因为可用的计算资源是有限的，所以优先级高的 kernel 任务可首先占用亲和资源来执行相应的运算，当高优先级下的 kernel 任务均被调度成功后，才可以对低优先级下的 kernel 任务进行任务调度。当同时存在不同优先级的 kernel 任务需要被调度时，根据所述第三寄存器中动态表格记录的当前可分配亲和资源，按照优先级由高到低的顺序，确定出每个优先级下的待调度任务为调度做准备。不同优先级下的待调度任务的亲和资源发生冲突时，优先调度优先级高的待调度任务，当不同优先级下的待调度任务的亲和资源不发生冲突时，可同时调度不同优先级下的待调度任务。可选择将不同等级的 kernel 任务需要占用的亲和资源都传输到任务调度系统的中枢模块，所述中枢模块被称作任务分离器（Task spliter，简称 TS）顶层，TS 顶层收到 kernel 任务需要占用的亲和资源后，根据优先级生成每个优先级对应的 kernel 任务亲和资源分配情况。具体的，将同一个

优先级下所有 kernel 任务的亲和资源进行资源掩码的位或操作，方便记录每一个优先级下 kernel 任务需占用的亲和资源的整体情况。当不同优先级下的 kernel 任务所占用的亲和资源不冲突时，即使是不同优先级下的 kernel 任务可以同时执行调度。比如，现在存在两个优先级的 kernel 任务均需要被调度，高优先级下包括 kernel1、kernel2、kernel3、kernel4 与 kernel5，资源掩码分别为：1110000000000000、0110000000000000、0001100000000000、0000010000000000 与 0000000110000000，将高优先级下的所有 kernel 任务的亲和资源进行资源掩码的位或操作后得到的结果时 1111110110000000，这代表高优先级下的所有 kernel 任务需要用到的亲和资源为 cluster10-cluster15、cluster8 与 cluster7。低优先级下包括 kernel6、kernel7 与 kernel8，资源掩码分别为：0000000011000000、0000001100100000 与 0000000000011100，将低优先级下的所有 kernel 任务的亲和资源进行资源掩码的位或操作后得到的结果时 0000001111111100，低优先级下的所有 kernel 任务需要用到的亲和资源为 cluster2-cluster9。由此可见，由此可见高优先级的 kernel 任务与低优先级的 kernel 任务之间，发生冲突的亲和资源只有 cluster8 与 cluster7，所以 cluster8 与 cluster7 会被高优先级的 kernel 任务优先占用，但是完成 kernel6 可以调用 cluster6，完成 kernel7 可以调用 cluster5 与 cluster9，所以此时高优先级下的 kernel 任务与低优先级的 kernel 任务之间的亲和资源整体是不冲突的，可以同时被调度，kernel6、kernel7 与 kernel8 不必等 kernel1、kernel2、kernel3、kernel4 与 kernel5 全部调度完后才执行被调度。本实例通过 TS 顶层对每个优先级下 kernel 任务之间的亲和资源的有效分配，在保障高优先级的 kernel 任务优先被调度的情况下，实现计算资源的最大使用程度，提高了芯片的计算效率。

在一种示例中，所述从每个优先级对应的任务中选取至少一个任务作为所述待调度任务，包括：

若所述优先级对应的任务中存在亲和资源有重合的不同任务，则将当前所述不同任务中权重最大的任务作为所述待调度任务，并增加所述不同任务中其它任务的权重。

结合场景示例，每个 kernel 任务自带权重的属性，权重属性的作用是在同等级的情况下，kernel 任务的权重高，就优先被调度。比如同一等级下的 kernel1、kernel2、kernel3 与 kernel4，他们的亲和资源均只有 cluster1，因为 kernel1 的初始权重最高，所以 kernel1 被首先调度成功，kernel2、kernel3 和 kernel4 与 kernel1 同等级的情况下，由于亲和资源冲突导致没有被调度成功，所以此时增加 kernel2、kernel3 和 kernel4 的权重，增加下次被调度成功的机率。

本实施例通过利用 kernel 任务的优先级，与同等级下 kernel 任务的权重属性，实现了 kernel 任务的有序执行，避免了 kernel 任务执行的杂乱。

实施例三

图 5 为本公开实施例三提供的一种任务调度装置的结构示意图，包括：

确定模块 51，用于从软件下发的任务中确定待调度任务，所述软件下发的任务设定有连续调度次数；

处理模块 52，用于将所述待调度任务拆分成多个子任务，并按照所述待调度任务的连续调度次数，

将所述多个子任务进行划分，得到所述待调度任务的至少一个区间任务；

处理模块 52，还用于针对所述待调度任务的每个区间任务，从所述待调度任务当前可见的资源中，为所述区间任务占据区间亲和资源，以及，将所述区间任务中的每个子任务连续调度至所述子任务对应的亲和资源，直至所述区间任务下的所有子任务均完成调度；其中，所述区间亲和资源包括所述区间任务中所有子任务对应的亲和资源的集合，其中被占据的资源对其它任务不可见。

结合场景示例：在软件运行的过程中，会生成大量需要被执行的任务，软件会将所述任务下发给芯片执行。软件下发的任务被称为 Kernel 任务，所述 Kernel 任务的粒度比较大，可以拆分成多个子任务。当软件给芯片以 Kernel 任务粒度的大小给芯片下发任务时，会同时下发所述 Kernel 任务对应的连续调度属性，即该 Kernel 任务可以连续调度的次数，作为一个参数写在 kernel 里，所述连续调度属性可通过软件灵活性设置。芯片在收到任务后，确定模块 51 会在大量需要被执行的 kernel 任务中确认出当前时刻下需要被执行的任务，也就是当前的待调度任务。处理模块 52 通过解析当前的待调度任务的连续调度参数控制连续调度的区间，从而让软件灵活的使用连续调度属性，所述连续调度指的是不间断的调度同一个连读调度区间中的子任务，直到一个连读调度区间中的子任务全部被调度完成。比如，若当前的待调度任务是一个可以被拆分成 10 个子任务的 kernel 任务，所述 10 个子任务分别为 job0-job9，并且软件指定的连读调度属性为 4，就以 4 个子任务为一个区间进行划分，比如 job0-job3 为一个区间，job4-job7 为一个区间，再将剩下的 job8 与 job9 作为一个区间，同一个区间中的子任务进行不间断的连续调度。

芯片完成每个调度任务除了需要相关的运算数据外，还需要占据一定的计算资源，所述计算资源可以理解为计算单元，芯片需要通过调度任务对应的运算数据在一定的计算单元内完成每个调度任务。在一个芯片中，一共有 16 个 IPU cluster，然后每个 IPU cluster 包含 4 个 IPU core，所述一个 IPU core 就是一个计算单元。每个 kernel 任务拆分出多个子任务，其中所述子任务具有两种类型，可以分为 block 任务与 UnionX 任务，由于 block 任务与 UnionX 任务的任务类型不同，所以需要占据的计算单元的多少不同，其中完成一个 block 任务需占用一个 IPU core，完成一个 UnionX 任务需要占用 X 个 IPU cluster，同一个 kernel 任务下的所有子任务的任务类型一致。

芯片中的所有计算单元并不全都符合完成某一个任务的条件，软件下发的 kernel 任务以及拆分出的子任务都有一些特定对应的一些计算单元来完成，可以完成某一任务的所有计算单元都是该任务对应的亲和资源。每个任务除了需要对应的运算数据外，还需要计算单元才可完成，但是所述需要的计算单元必须是任务对应的亲和资源。

为了完成一个区间中所有子任务的连续调度，当一个区间中的首个子任务被调度成功后，那么所述区间中每个子任务对应的所有亲和资源对其他 kernel 任务不可见，其他 kernel 任务的子任务不可以再占用已不可见的计算单元，所述区间中每个子任务对应的所有亲和资源只为所述区间中的子任务提供计算单元。同一个 kernel 任务下的所有子任务对应的亲和资源是一致的，比如，当上述 job0-job3 这一个区间中的首个任务，也就是 job0 被调度成功后，假如 job0 所属的 kernel 下的 job 任务需要的亲和资源有 IPU

cluster1、IPU cluster2 与 IPU cluster3，那当 job0 被调度成功后，将占据 IPU cluster1、IPU cluster2 与 IPU cluster3，并对其他 kernel 任务不可见。往往一个任务对应的亲和资源都比完成该任务需要的实际资源要大很多，比如完成 job0 只需要用到 IPU cluster1 中的一个计算单元，就是说完成 job0 只需要占用 IPU cluster1 中的一个 IPU core，那么在 IPU cluster1 中剩余的其他 3 个 IPU core 将作为同一区间中其他子任务可供选择的亲和资源。

本示例处理模块通过将多个子任务按区间进行划分，并通过占据所述区间中所有子任务的亲和资源对其他 kernel 任务不可见，连续调度所述区间中每个子任务，直至所述区间下的所有子任务均完成调度。因为来自同一个 kernel 任务的处理数据会有强相关性，比如指令地址、数据地址、参数等信息会有大量的重复，所以当计算单元执行来自同一个 kernel 任务的子任务时，可以有效提升访问 cache 的命中率，减少了数据在缓存器中的切换次数，从而有效提升芯片计算效率。

在一种示例中，处理模块 52，还用于若所述区间任务中的所有子任务均完成调度，则释放所述区间任务占据的所述区间亲和资源，其中被释放的资源对其它任务可见。

当该区间中每个子任务均被调度成功到亲和资源上并完成每个子任务的计算后，将释放被占据的资源对其他 kernel 任务可见。比如当 job0-job3 均被成调度功到对应的亲和资源上，并在计算资源上完成 job0-job3 的计算后，会释放所述 job0-job3 的亲和资源 IPU cluster1、IPU cluster2 与 IPU cluster3 对其他 kernel 任务可见。

在一种示例中，处理模块 52，具体用于获取所述区间任务中子任务对应的资源掩码，所述子任务对应的资源掩码表征所述子任务对应的亲和资源；

处理模块 52，具体还用于将所述待调度任务当前可见的资源中所述掩码表征的资源，作为所述区间亲和资源，为所述区间任务占用所述区间亲和资源。

在一个芯片上一共有 16 个 IPU cluster，每个 Kernel 任务都有各自对应的亲和资源，每个 Kernel 任务对应的亲和资源中包括的 IPU cluster 可用资源掩码来对应，所述资源掩码可以采用二进制数字来表征。比如针对 job0 所属的 Kernel 任务来说，每个 IPU cluster 若是 job0 的亲和资源可用 1 来表示，若不是 job0 的亲和资源可用 0 来表示。假如 job0 的亲和资源有 IPU cluster1、IPU cluster2 与 IPU cluster3，那么只有 IPU cluster1、IPU cluster2 与 IPU cluster3 为 1，其他 IPU cluster 均为 0，那么用来表征 job0 的亲和资源的资源掩码为 0000000000001110，资源掩码的数位从右到左依次代表 IPU cluster0-IPU cluster15 对 Kernel 任务亲和资源的表征结果。本实例采用资源掩码的方式表征子任务对应的亲和资源，可在调度子任务时准确调度到每个子任务对应的亲和资源上。

在一种示例中，处理模块 52，具体还用于通过解析所述任务调度指令，获得所述连续调度次数并记录在第一寄存器中；

处理模块 52，具体还用于基于第一更新条件，更新第二寄存器中的数据，所述第二寄存器中的数据表征当前已连续调度的次数；其中，所述第一更新条件包括：若当前已成功连续调度的次数达到所述第

一寄存器中的连续调度次数，或者执行所述将所述待调度任务拆分成多个子任务的步骤后，或者收到针对所述待调度任务的终止调度指令，或者当前达到最大并行度，则对所述第二寄存器清零；否则，每次完成子任务的调度后，将所述第二寄存器执行加 1 处理；

处理模块 52，具体还用于基于第二更新条件，更新第三寄存器中的数据，所述第三寄存器中的数据表征当前需为所述区间任务占据的区间亲和资源；其中，所述第二更新条件包括：若当前已成功连续调度的次数达到所述第一寄存器中的连续调度次数，或者执行所述将所述待调度任务拆分成多个子任务的步骤后，或者收到针对所述待调度任务的终止调度指令，或者当前达到最大并行度，则清除所述第三寄存器中的数据；否则，每次完成子任务的调度后，从所述第三寄存器中删除该子任务的亲和资源。

根据场景示例，执行连续调度的过程中，当满足连续调度停止的条件时，连续调度将停止，在此之前，需要首先解析出软件下发的 kernel 任务的连续调度参数，当解析出所述参数后，将其记录在芯片中的第一寄存器中，以供后续参考。除了第一寄存器，芯片中还存在第二寄存器与第三寄存器，所述第二寄存器主要的任务是维护一个表示当前连续调度次数的表格，在连续调度的过程中，基于第一更新条件更新所述表格，所述第一更新条件除了包括次数清零条件，好包括若子任务每成功调度一次，则将表格中的次数加 1。所述清零条件包括：条件 1：当前已成功连续调度的次数达到所述第一寄存器中的连续调度次数；条件 2：执行所述将待调度任务被拆分成多个子任务的步骤后；条件 3：收到针对所述待调度任务的终止调度指令；条件 4：当前达到最大并行度。

所述条件 1 指的是连续调度的次数达到所述第一寄存器中记录的连续调度参数，当前连续调度区间中的所有子任务均被调度成功，所以对所述第二寄存器中维护的表格清零以方便记录下次连续调度的次数。所述条件 2 指的是待调度任务刚被拆分成多个子任务需要进行连续调度，相当于初始化清零。所述条件 3 指的是当前连续调度的过程中，存在子任务不惜要被执行，则清楚目前连续调度的次数以方便记录下次连续调度的次数。所述条件 4，最大并行度是指一个 kernel 最多占用的资源数量，若一个 kernel 只允许被占用 4 个 cluster，那 4 就是他的最大并行度，当目前同时执行的子任务达到对的 kernel 的最大并行度，则清楚目前连续调度的次数以方便记录下次连续调度的次数。

所述第三寄存器主要的任务是记录当前待调度任务可占用的资源，并生成一个动态表格，在连续调度的过程中，基于第二更新条件更新所述动态表格，所述第二更新条件除了包括清零条件外，当连读调度区间内每个子任务被调度成功后，从所述第三寄存器中删除该子任务的亲和资源。具体的，比如 job0-job3 这一个区间中子任务对应的亲和资源为 IPU cluster1、IPU cluster2 与 IPU cluster3，当 job0 调度成功后，此时所述第三寄存器中动态表格将删除所述 IPU cluster1、IPU cluster2 与 IPU cluster3，所述 IPU cluster1、IPU cluster2 与 IPU cluster3 将对其他待调度任务不可见。

需要说明的是，若执行 job0 需要占用 IPU cluster1 中的一个 IPU core，当 job0 调度成功后，与所述 job0 同一个区间内的其他子任务不会再占用所述 job0 占用的 IPU core，但是当 job0 被执行完毕，不需要再占用计算资源时，可对同一个区间内的其他子任务释放出所占用的 IPU core。

在一种示例中，若是在此区间内的所有子任务在全部调度成功之前，之前已被调度成功的子任务已完成，则执行该子任务所占用的亲和资源会被闲置出来，所以在动态表格下一次更新后，会显示出被闲置的亲和资源。比如在 job3 调度成功后，job0 已完成，那被 job0 占用的 IPU cluster1 中的一个 IPU core 会显示在此时的动态表格中。

本示例通过维护表示当前连续调度次数的表格与当前子任务可占用的资源的动态表格，直观的表现出当前连续调度的次数与资源占用情况，避免造成资源的浪费与冲突。

在一种示例中，处理模块 52，具体还用于若第二寄存器中的数据未达到第一寄存器中的连续调度次数，则确定所述区间任务中未完成调度的子任务；

处理模块 52，具体还用于从所述未完成调度的子任务中选取待调度的子任务，将所述待调度的子任务调度至所述子任务对应的亲和资源，直至所述区间任务中所有子任务均完成调度。

当连续调度的任务在正常执行时，所述第二寄存器中维护的表格是表示当前连续调度次数的表格，当表格中记录的次数没有达到所述第一寄存器中记录的参数时，说明当权连续调度区间中仍有没有被调度，则从没有被调度的子任务中选取待调度子任务，将当前的待调度子任务调度到对应的亲和资源上，直到当前连续调度区间中的子任务都被调度完成。本示例通过参考表示当前连续调度次数的表格与当前子任务可占用的资源的动态表格，准确的选取待调度子任务与当前可调度的亲和资源，避免造成资源的浪费与冲突。

在一种示例中，所述处理模块，具体还用于根据当前的可用资源，按照优先级自高向低，从每个优先级对应的任务中选取至少一个任务作为所述待调度任务，直至当前的可用资源不包括任意任务对应的亲和资源；其中，不同待调度任务对应的亲和资源不同，所述可用资源包括除当前的所有待调度任务对应的亲和资源以外的资源。

结合场景示例，当同时存在多个 kernel 任务需要被调度时，因为可用的计算资源是有限的，所以优先级高的 kernel 任务可首先占用亲和资源来执行相应的运算，当高优先级下的 kernel 任务均被调度成功后，才可以对低优先级下的 kernel 任务进行任务调度。当同时存在不同优先级的 kernel 任务需要被调度时，根据所述第三寄存器中动态表格记录的当前可分配亲和资源，按照优先级由高到低的顺序，确定出每个优先级下的待调度任务为调度做准备。不同优先级下的待调度任务的亲和资源发生冲突时，优先调度优先级高的待调度任务，当不同优先级下的待调度任务的亲和资源不发生冲突时，可同时调度不同优先级下的待调度任务。可选择将不同等级的 kernel 任务需要占用的亲和资源都传输到任务调度系统的中枢模块，所述中枢模块被称作任务分离器（Task spliter，简称 TS）顶层，TS 顶层收到 kernel 任务需要占用的亲和资源后，根据优先级生成每个优先级对应的 kernel 任务亲和资源分配情况。具体的，将同一个优先级下所有 kernel 任务的亲和资源进行资源掩码的位或操作，方便记录每一个优先级下 kernel 任务需占用的亲和资源的整体情况。当不同优先级下的 kernel 任务所占用的亲和资源不冲突时，即使是不同优先级下的 kernel 任务可以同时执行调度。比如，现在存在两个优先级的 kernel 任务均需要被调度，高优

先级下包括 kernel1、kernel2、kernel3、kernel4 与 kernel5，资源掩码分别为：1110000000000000、0110000000000000、0001100000000000、0000010000000000 与 0000000110000000，将高优先级下的所有 kernel 任务的亲和资源进行资源掩码的位或操作后得到的结果时 1111110110000000，这代表高优先级下的所有 kernel 任务需要用到的亲和资源为 cluster10-cluster15、cluster8 与 cluster7。低优先级下包括 kernel6、kernel7 与 kernel8，资源掩码分别为：0000000011000000、0000001100100000 与 0000000000011100，将低优先级下的所有 kernel 任务的亲和资源进行资源掩码的位或操作后得到的结果时 0000001111111100，低优先级下的所有 kernel 任务需要用到的亲和资源为 cluster2-cluster9。由此可见，由此可见高优先级的 kernel 任务与低优先级的 kernel 任务之间，发生冲突的亲和资源只有 cluster8 与 cluster7，所以 cluster8 与 cluster7 会被高优先级的 kernel 任务优先占用，但是完成 kernel6 可以调用 cluster6，完成 kernel7 可以调用 cluster5 与 cluster9，所以此时高优先级下的 kernel 任务与低优先级的 kernel 任务之间的亲和资源整体是不冲突的，可以同时被调度，kernel6、kernel7 与 kernel8 不必等 kernel1、kernel2、kernel3、kernel4 与 kernel5 全部调度完后才执行被调度。

本实例通过 TS 顶层对每个优先级下 kernel 任务之间的亲和资源的有效分配，在保障高优先级的 kernel 任务优先被调度的情况下，实现计算资源的最大使用程度，提高了芯片的计算效率。

在一种示例中，所述处理模块，具体还用于若所述优先级对应的任务中存在亲和资源有重合的不同任务，则将当前所述不同任务中权重最大的任务作为所述待调度任务，并增加所述不同任务中其它任务的权重。

结合场景示例，每个 kernel 任务自带权重的属性，权重属性的作用是在同等级的情况下，kernel 任务的权重高，就优先被调度。比如同一等级下的 kernel1、kernel2、kernel3 与 kernel4，他们的亲和资源均只有 cluster1，因为 kernel1 的初始权重最高，所以 kernel1 被首先调度成功，kernel2、kernel3 和 kernel4 与 kernel1 同等级的情况下，由于亲和资源冲突导致没有被调度成功，所以此时增加 kernel2、kernel3 和 kernel4 的权重，增加下次被调度成功的机率。

本实施例通过利用 kernel 任务的优先级，与同等级下 kernel 任务的权重属性，实现了 kernel 任务的有序执行，避免了 kernel 任务执行的杂乱。

实施例四

图 6 是根据一示例性实施例示出的一种中央控制单元的装置框图，该装置可以是移动电话，计算机，数字广播终端，消息收发设备，游戏控制台，平板设备，医疗设备，健身设备，个人数字助理等。

装置 800 可以包括以下一个或多个组件：处理组件 802，存储器 804，电源组件 806，多媒体组件 808，音频组件 810，输入/输出（I/O）接口 812，传感器组件 814，以及通信组件 816。

处理组件 802 通常控制装置 800 的整体操作，诸如与显示，电话呼叫，数据通信，相机操作和记录操作相关联的操作。处理组件 802 可以包括一个或多个处理器 820 来执行指令，以完成上述的方法的全

部或部分步骤。此外，处理组件 802 可以包括一个或多个模块，便于处理组件 802 和其他组件之间的交互。例如，处理组件 802 可以包括多媒体模块，以方便多媒体组件 808 和处理组件 802 之间的交互。

存储器 804 被配置为存储各种类型的数据以支持在装置 800 的操作。这些数据的示例包括用于在装置 800 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 804 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。

电源组件 806 为装置 800 的各种组件提供电力。电源组件 806 可以包括电源管理系统，一个或多个电源，及其他与为装置 800 生成、管理和分配电力相关联的组件。

多媒体组件 808 包括在所述装置 800 和用户之间的提供一个输出接口的屏幕。在一些实施例中，屏幕可以包括液晶显示器（LCD）和触摸面板（TP）。如果屏幕包括触摸面板，屏幕可以被实现为触摸屏，以接收来自用户的输入信号。触摸面板包括一个或多个触摸传感器以感测触摸、滑动和触摸面板上的手势。所述触摸传感器可以不仅感测触摸或滑动动作的边界，而且还检测与所述触摸或滑动操作相关的持续时间和压力。在一些实施例中，多媒体组件 808 包括一个前置摄像头和/或后置摄像头。当装置 800 处于操作模式，如拍摄模式或视频模式时，前置摄像头和/或后置摄像头可以接收外部的多媒体数据。每个前置摄像头和后置摄像头可以是一个固定的光学透镜系统或具有焦距和光学变焦能力。

音频组件 810 被配置为输出和/或输入音频信号。例如，音频组件 810 包括一个麦克风（MIC），当装置 800 处于操作模式，如呼叫模式、记录模式和语音识别模式时，麦克风被配置为接收外部音频信号。所接收的音频信号可以被进一步存储在存储器 804 或经由通信组件 816 发送。在一些实施例中，音频组件 810 还包括一个扬声器，用于输出音频信号。

I/O 接口 812 为处理组件 802 和外围接口模块之间提供接口，上述外围接口模块可以是键盘，点击轮，按钮等。这些按钮可包括但不限于：主页按钮、音量按钮、启动按钮和锁定按钮。

传感器组件 814 包括一个或多个传感器，用于为装置 800 提供各个方面的状态评估。例如，传感器组件 814 可以检测到装置 800 的打开/关闭状态，组件的相对定位，例如所述组件为装置 800 的显示器和小键盘，传感器组件 814 还可以检测装置 800 或装置 800 一个组件的位置改变，用户与装置 800 接触的存在或不存在，装置 800 方位或加速/减速和装置 800 的温度变化。传感器组件 814 可以包括接近传感器，被配置用来在没有任何的物理接触时检测附近物体的存在。传感器组件 814 还可以包括光传感器，如 CMOS 或 CCD 图像传感器，用于在成像应用中使用。在一些实施例中，该传感器组件 814 还可以包括加速度传感器，陀螺仪传感器，磁传感器，压力传感器或温度传感器。

通信组件 816 被配置为便于装置 800 和其他设备之间有线或无线方式的通信。装置 800 可以接入基于通信标准的无线网络，如 WiFi，2G 或 3G，或它们的组合。在一个示例性实施例中，通信组件 816 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 816 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。

在示例性实施例中，装置 800 可以被一个或多个应用专用集成电路（ASIC）、数字信号处理器（DSP）、数字信号处理设备（DSPD）、可编程逻辑器件（PLD）、现场可编程门阵列（FPGA）、控制器、微控制器、微处理器或其他电子元件实现，用于执行上述方法。

在示例性实施例中，还提供了一种包括指令的非临时性计算机可读存储介质，例如包括指令的存储器 804，上述指令可由装置 800 的处理器 820 执行以完成上述方法。例如，所述非临时性计算机可读存储介质可以是 ROM、随机存取存储器（RAM）、CD-ROM、磁带、软盘和光数据存储设备等。

实施例五

图 7 为本公开实施例五中提供的一种电子设备的结构示意图，如图所示，该电子设备包括：

处理器（processor）291，电子设备还包括了存储器（memory）292；还可以包括通信接口（Communication Interface）293 和总线 294。其中，处理器 291、存储器 292、通信接口 293、可以通过总线 294 完成相互间的通信。通信接口 293 可以用于信息传输。处理器 291 可以调用存储器 294 中的逻辑指令，以执行上述实施例的方法。

此外，上述的存储器 292 中的逻辑指令可以通过软件功能单元的形式实现并作为独立的产品销售或使用时，可以存储在一个计算机可读取存储介质中。

存储器 292 作为一种计算机可读存储介质，可用于存储软件程序、计算机可执行程序，如本公开实施例中的方法对应的程序指令/模块。处理器 291 通过运行存储在存储器 292 中的软件程序、指令以及模块，从而执行功能应用以及数据处理，即实现上述方法实施例中的方法。

存储器 292 可包括存储程序区和存储数据区，其中，存储程序区可存储操作系统、至少一个功能所需的应用程序；存储数据区可存储根据终端设备的使用所创建的数据等。此外，存储器 292 可以包括高速随机存取存储器，还可以包括非易失性存储器。

本公开实施例提供一种非临时性计算机可读存储介质，所述计算机可读存储介质中存储有计算机执行指令，所述计算机执行指令被处理器执行时用于实现如前述实施例所述的方法。

本公开实施例提供一种计算机程序产品，包括计算机程序，所述计算机程序被处理器执行时实现如前述实施例所述的方法。

本领域技术人员在考虑说明书及实践这里公开的发明后，将容易想到本公开的其它实施方案。本公开旨在涵盖本公开的任何变型、用途或者适应性变化，这些变型、用途或者适应性变化遵循本公开的一般性原理并包括本公开未公开的本技术领域中的公知常识或惯用技术手段。说明书和实施例仅被视为示例性的，本公开的真正范围和精神由下面的权利要求书指出。

应当理解的是，本公开并不局限于上面已经描述并在附图中示出的精确结构，并且可以在不脱离其

范围进行各种修改和改变。本公开的范围仅由所附的权利要求书来限制。

权 利 要 求 书

1、一种任务调度方法，其特征在于，包括：

从软件下发的任务中确定待调度任务，所述软件下发的任务设定有连续调度次数；

将所述待调度任务拆分成多个子任务，并按照所述待调度任务的连续调度次数，将所述多个子任务进行划分，得到所述待调度任务的至少一个区间任务；

针对所述待调度任务的每个区间任务，从所述待调度任务当前可见的资源中，为所述区间任务占据区间亲和资源，以及，将所述区间任务中的每个子任务连续调度至所述子任务对应的亲和资源，直至所述区间任务下的所有子任务均完成调度；其中，所述区间亲和资源包括所述区间任务中所有子任务对应的亲和资源的集合，其中被占据的资源对其它任务不可见。

2、根据权利要求 1 所述的方法，其特征在于，所述方法还包括：

若所述区间任务中的所有子任务均完成调度，则释放所述区间任务占据的所述区间亲和资源，其中被释放的资源对其它任务可见。

3、根据权利要求 1 或 2 所述的方法，其特征在于，从所述待调度任务当前可见的资源中，为所述区间任务占据区间亲和资源，包括：

获取所述区间任务中子任务对应的资源掩码，所述子任务对应的资源掩码表征所述子任务对应的亲和资源；

将所述待调度任务当前可见的资源中所述掩码表征的资源，作为所述区间亲和资源，为所述区间任务占用所述区间亲和资源。

4、根据权利要求 1-3 任一项所述的方法，其特征在于，所述方法还包括：

通过解析所述任务调度指令，获得所述连续调度次数并记录在第一寄存器中；

基于第一更新条件，更新第二寄存器中的数据，所述第二寄存器中的数据表征当前已连续调度的次数；其中，所述第一更新条件包括：若当前已成功连续调度的次数达到所述第一寄存器中的连续调度次数，或者执行所述将所述待调度任务拆分成多个子任务的步骤后，或者收到针对所述待调度任务的终止调度指令，或者当前达到最大并行度，则对所述第二寄存器清零；否则，每次完成子任务的调度后，将所述第二寄存器执行加 1 处理；

基于第二更新条件，更新第三寄存器中的数据，所述第三寄存器中的数据表征当前需为所述区间任务占据的区间亲和资源；其中，所述第二更新条件包括：若当前已成功连续调度的次数达到所述第一寄存器中的连续调度次数，或者执行所述将所述待调度任务拆分成多个子任务的步骤后，或者收到针对所述待调度任务的终止调度指令，或者当前达到最大并行度，则清除所述第三寄存器中的数据；否则，每次完成子任务的调度后，从所述第三寄存器中删除该子任务的亲和资源。

5、根据权利要求 1-4 任一项所述的方法，其特征在于，将所述区间任务中的每个子任务连续调度至所述子任务对应的亲和资源，包括：

若第二寄存器中的数据未达到第一寄存器中的连续调度次数，则确定所述区间任务中未完成调度的子任务；

从所述未完成调度的子任务中选取待调度的子任务，将所述待调度的子任务调度至所述子任务对应的亲和资源，直至所述区间任务中所有子任务均完成调度。

6、根据权利要求 1-5 任一项所述的方法，其特征在于，所述软件下发的任务具有优先级；所述从软件下发的任务中确定待调度任务，包括：

根据当前的可用资源，按照优先级自高向低，从每个优先级对应的任务中选取至少一个任务作为所述待调度任务，直至当前的可用资源不包括任意任务对应的亲和资源；其中，不同待调度任务对应的亲和资源不同，所述可用资源包括除当前的所有待调度任务对应的亲和资源以外的资源。

7、根据权利要求 6 所述的方法，其特征在于，所述从每个优先级对应的任务中选取至少一个任务作为所述待调度任务，包括：

若所述优先级对应的任务中存在亲和资源有重合的不同任务，则将当前所述不同任务中权重最大的任务作为所述待调度任务，并增加所述不同任务中其它任务的权重。

8、一种任务调度装置，其特征在于，包括：

确定模块，用于从软件下发的任务中确定待调度任务，所述软件下发的任务设定有连续调度次数；

处理模块，用于将所述待调度任务拆分成多个子任务，并按照所述待调度任务的连续调度次数，将所述多个子任务进行划分，得到所述待调度任务的至少一个区间任务；

处理模块，还用于针对所述待调度任务的每个区间任务，从所述待调度任务当前可见的资源中，为所述区间任务占据区间亲和资源，以及，将所述区间任务中的每个子任务连续调度至所述子任务对应的亲和资源，直至所述区间任务下的所有子任务均完成调度；其中，所述区间亲和资源包括所述区间任务中所有子任务对应的亲和资源的集合，其中被占据的资源对其它任务不可见。

9、根据权利要求 8 所述的装置，其特征在于，

所述处理模块，还用于若所述区间任务中的所有子任务均完成调度，则释放所述区间任务占据的所述区间亲和资源，其中被释放的资源对其它任务可见。

10、根据权利要求 8 或 9 所述的装置，其特征在于，

所述处理模块，具体用于获取所述区间任务中子任务对应的资源掩码，所述子任务对应的资源掩码表征所述子任务对应的亲和资源；

所述处理模块，具体还用于将所述待调度任务当前可见的资源中所述掩码表征的资源，作为所述区间亲和资源，为所述区间任务占用所述区间亲和资源。

11、根据权利要求 8-10 任一项所述的装置，其特征在于，

所述处理模块，具体还用于通过解析所述任务调度指令，获得所述连续调度次数并记录在第一寄存器中；

所述处理模块，具体还用于基于第一更新条件，更新第二寄存器中的数据，所述第二寄存器中的数据表征当前已连续调度的次数；其中，所述第一更新条件包括：若当前已成功连续调度的次数达到所述第一寄存器中的连续调度次数，或者执行所述将所述待调度任务拆分成多个子任务的步骤后，或者收到针对所述待调度任务的终止调度指令，或者当前达到最大并行度，则对所述第二寄存器清零；否则，每次完成子任务的调度后，将所述第二寄存器执行加 1 处理；

所述处理模块，具体还用于基于第二更新条件，更新第三寄存器中的数据，所述第三寄存器中的数据表征当前需为所述区间任务占据的区间亲和资源；其中，所述第二更新条件包括：若当前已成功连续调度的次数达到所述第一寄存器中的连续调度次数，或者执行所述将所述待调度任务拆分成多个子任务的步骤后，或者收到针对所述待调度任务的终止调度指令，或者当前达到最大并行度，则清除所述第三寄存器中的数据；否则，每次完成子任务的调度后，从所述第三寄存器中删除该子任务的亲和资源。

12、根据权利要求 8-11 任一项所述的装置，其特征在于，

所述处理模块，具体还用于若第二寄存器中的数据未达到第一寄存器中的连续调度次数，则确定所述区间任务中未完成调度的子任务；

所述处理模块，具体还用于从所述未完成调度的子任务中选取待调度的子任务，将所述待调度的子任务调度至所述子任务对应的亲和资源，直至所述区间任务中所有子任务均完成调度。

13、根据权利要求 8-12 任一项所述的装置，其特征在于，

所述处理模块，具体还用于根据当前的可用资源，按照优先级自高向低，从每个优先级对应的任务中选取至少一个任务作为所述待调度任务，直至当前的可用资源不包括任意任务对应的亲和资源；其中，不同待调度任务对应的亲和资源不同，所述可用资源包括除当前的所有待调度任务对应的亲和资源以外的资源。

14、根据权利要求 13 所述的装置，其特征在于，

所述处理模块，具体还用于若所述优先级对应的任务中存在亲和资源有重合的不同任务，则将当前所述不同任务中权重最大的任务作为所述待调度任务，并增加所述不同任务中其它任务的权重。

15、一种电子设备，其特征在于，包括：处理器，以及与所述处理器通信连接的存储器；

所述存储器存储计算机执行指令；

所述处理器执行所述存储器存储的计算机执行指令，以实现如权利要求 1-7 中任一项所述的方法。

16、一种计算机可读存储介质，其特征在于，所述计算机可读存储介质中存储有计算机执行指令，所述计算机执行指令被处理器执行时用于实现如权利要求 1-7 中任一项所述的方法。

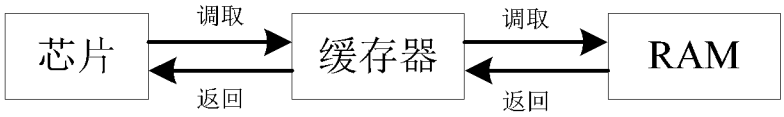
芯片
调取
返回
缓存器
调取
返回
RAM

图 1

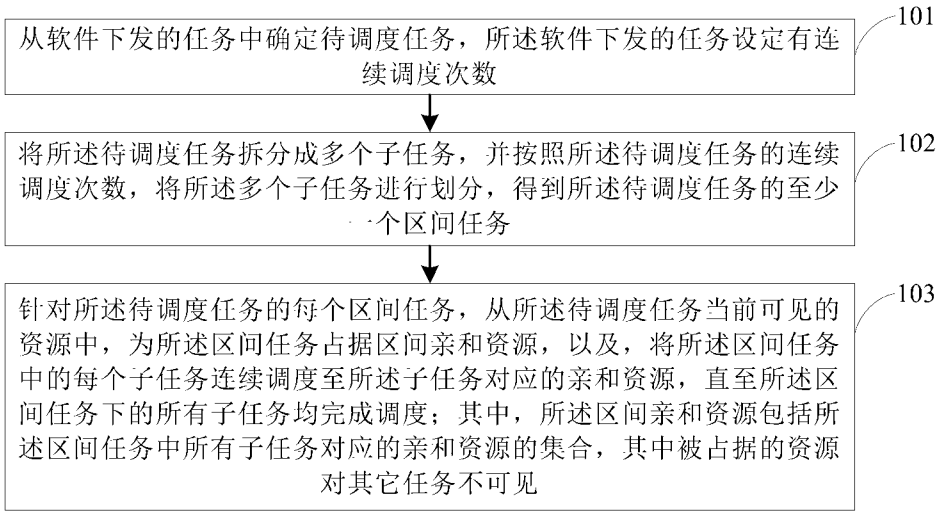
从软件下发的任务中确定待调度任务，所述软件下发的任务设定有连续调度次数
101
将所述待调度任务拆分成多个子任务，并按照所述待调度任务的连续调度次数，将所述多个子任务进行划分，得到所述待调度任务的至少一个区间任务
102
针对所述待调度任务的每个区间任务，从所述待调度任务当前可见的资源中，为所述区间任务占据区间亲和资源，以及，将所述区间任务中的每个子任务连续调度至所述子任务对应的亲和资源，直至所述区间任务下的所有子任务均完成调度；其中，所述区间亲和资源包括所述区间任务中所有子任务对应的亲和资源的集合，其中被占据的资源对其它任务不可见
103

图 2

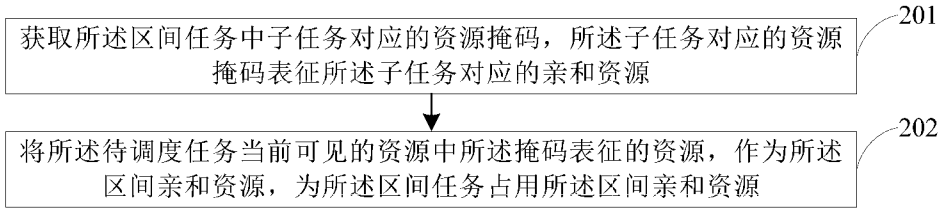
获取所述区间任务中子任务对应的资源掩码，所述子任务对应的资源掩码表征所述子任务对应的亲和资源
201
将所述待调度任务当前可见的资源中所述掩码表征的资源，作为所述区间亲和资源，为所述区间任务占用所述区间亲和资源
202

图 3

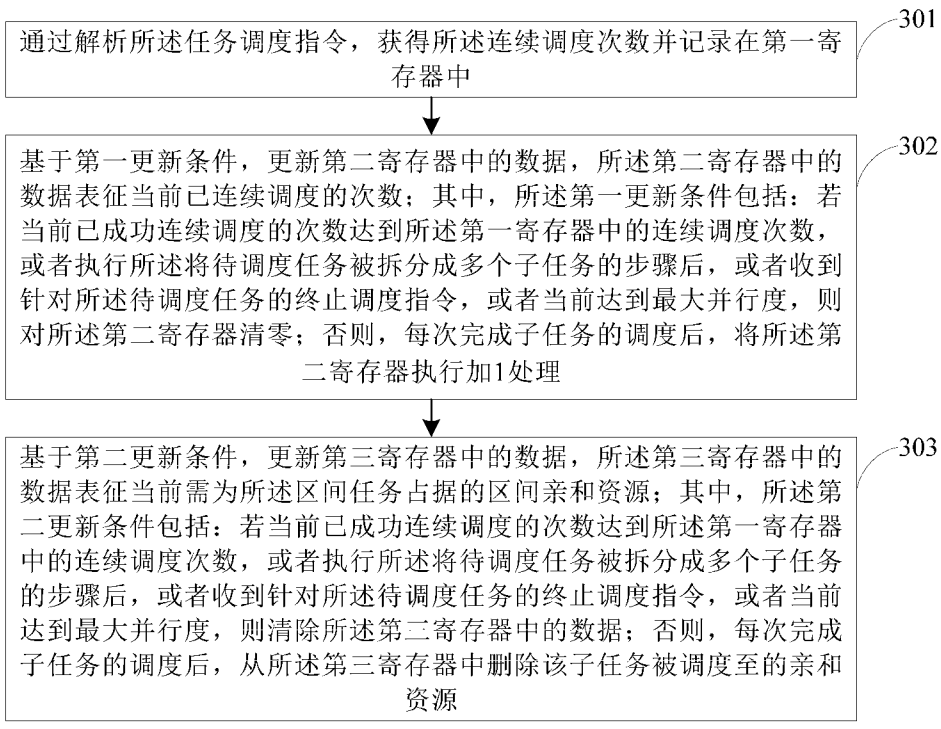
通过解析所述任务调度指令，获得所述连续调度次数并记录在第一寄存器中
301
基于第一更新条件，更新第二寄存器中的数据，所述第二寄存器中的数据表征当前已连续调度的次数；其中，所述第一更新条件包括：若当前已成功连续调度的次数达到所述第一寄存器中的连续调度次数，或者执行所述将待调度任务被拆分成多个子任务的步骤后，或者收到针对所述待调度任务的终止调度指令，或者当前达到最大并行度，则对所述第二寄存器清零；否则，每次完成子任务的调度后，将所述第二寄存器执行加1处理
302
基于第二更新条件，更新第三寄存器中的数据，所述第三寄存器中的数据表征当前需为所述区间任务占据的区间亲和资源；其中，所述第二更新条件包括：若当前已成功连续调度的次数达到所述第一寄存器中的连续调度次数，或者执行所述将待调度任务被拆分成多个子任务的步骤后，或者收到针对所述待调度任务的终止调度指令，或者当前达到最大并行度，则清除所述第三寄存器中的数据；否则，每次完成子任务的调度后，从所述第三寄存器中删除该子任务被调度至的亲和资源
303

图 4

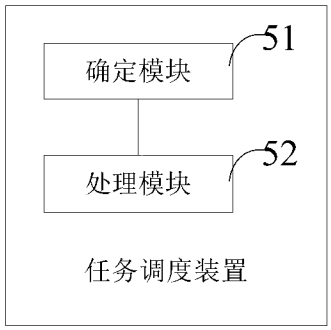
确定模块
51
处理模块
52
任务调度装置

图 5

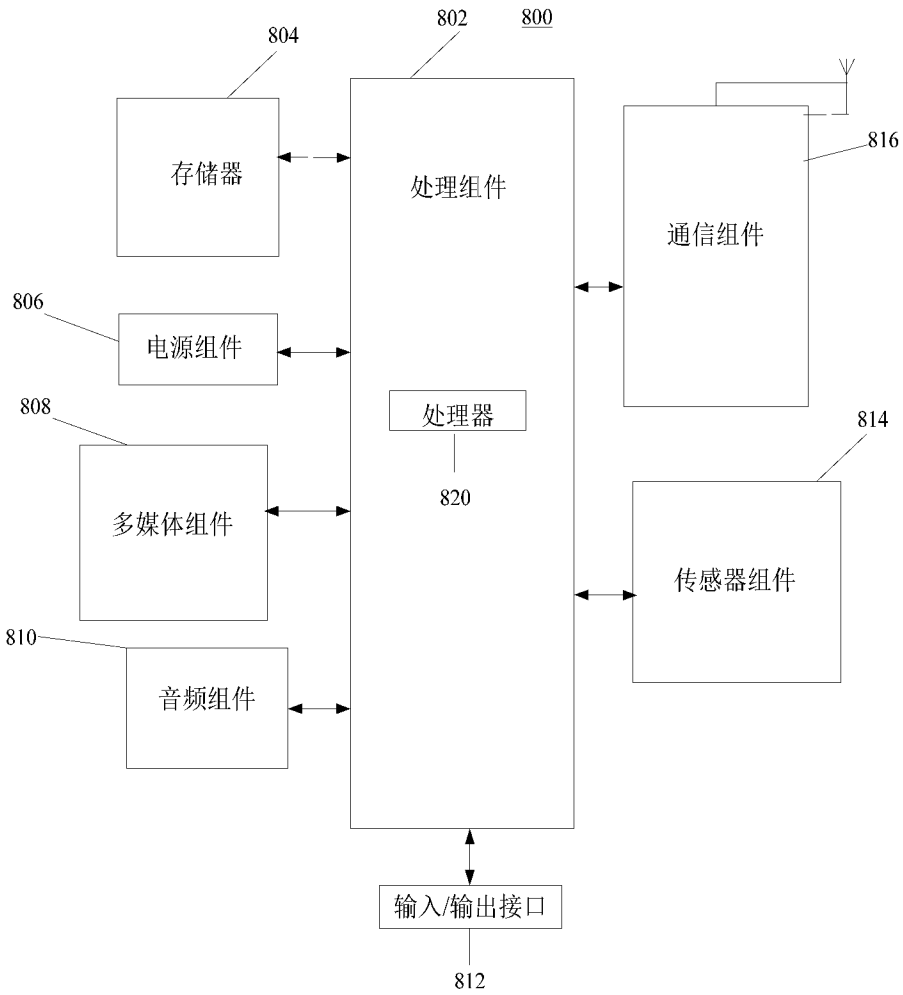
802
800
804
存储器
处理组件
通信组件
816
806
电源组件
处理器
820
808
多媒体组件
814
传感器组件
810
音频组件
输入/输出接口
812

图 6

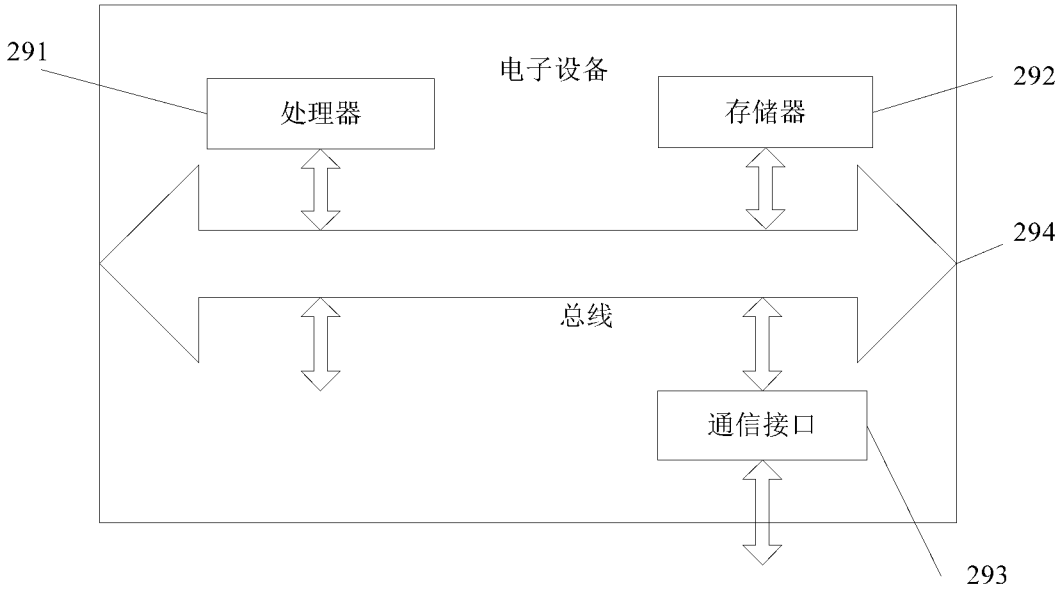

图 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2023/104517

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/48(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

VEN, CNABS, CNTXT, WOTXT, EPTXT, USTXT, CNKI, IEEE: 连续, 持续, 不间断, 子任务, 分配, 调度, 调配, 分派, 分发, 指派, 划分, 拆, 分割, 分成, 切割, 分为, 分成, 亲和, 优先, continue, task, allocate, schedule, split, cut, apart, affinity, prior

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2021216557 A1 (EMC IP HOLDING COMPANY L.L.C.) 15 July 2021 (2021-07-15) description, paragraphs 18-96	1-16
Y	CN 105260244 A (BEIJING QIYI CENTURY SCIENCE & TECHNOLOGY CO., LTD.) 20 January 2016 (2016-01-20) description, paragraphs 65-123	1-16
Y	US 5761512 A (IBM) 02 June 1998 (1998-06-02) description, columns 2-7	1-16
Y	US 2010153963 A1 (KAKARLAMUDI, Subbarao et al.) 17 June 2010 (2010-06-17) description, paragraph [0046]	1-16
Y	WO 2022067531 A1 (SHENZHEN UNIVERSITY) 07 April 2022 (2022-04-07) description, pages 6-11	1-16
A	US 2018115496 A1 (ADVANCED MICRO DEVICES, INC.) 26 April 2018 (2018-04-26) entire document	1-16
A	CN 114924858 A (BANK OF CHINA CO., LTD.) 19 August 2022 (2022-08-19) entire document	1-16

☑ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"D" document cited by the applicant in the international application
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
14 August 2023	**22 August 2023**

Name and mailing address of the ISA/CN	Authorized officer
China National Intellectual Property Administration (ISA/CN) **China No. 6, Xitucheng Road, Jimenqiao, Haidian District, Beijing 100088**	Telephone No.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2009300623 A1 (BANSAL, Nikhil et al.) 03 December 2009 (2009-12-03) entire document	1-16
A	CN 114741207 A (ZHEJIANG LAB) 12 July 2022 (2022-07-12) entire document	1-16

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2023/104517

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 2021216557 A1	15 July 2021	US 11436231 B2	06 September 2022
CN 105260244 A	20 January 2016	None	
US 5761512 A	02 June 1998	GB 9626242 D0	05 February 1997
		GB 2308906 A	09 July 1997
		GB 2308906 B	20 October 1999
US 2010153963 A1	17 June 2010	US 9390130 B2	12 July 2016
WO 2022067531 A1	07 April 2022	CN 112214319 A	12 January 2021
US 2018115496 A1	26 April 2018	JP 2019537104 A	19 December 2019
		EP 3529697 A1	28 August 2019
		KR 20190070915 A	21 June 2019
		WO 2018075131 A1	26 April 2018
		CN 109791507 A	21 May 2019
		IN 201917006051 A	26 April 2019
CN 114924858 A	19 August 2022	None	
US 2009300623 A1	03 December 2009	US 8458720 B2	04 June 2013
CN 114741207 A	12 July 2022	None	

国际检索报告

国际申请号
PCT/CN2023/104517

A. 主题的分类

G06F 9/48(2006.01)i

按照国际专利分类(IPC)或者同时按照国家分类和IPC两种分类

B. 检索领域

检索的最低限度文献(标明分类系统和分类号)

IPC：G06F

包含在检索领域中的除最低限度文献以外的检索文献

在国际检索时查阅的电子数据库(数据库的名称，和使用的检索词（如使用））

VEN,CNABS,CNTXT,WOTXT,EPTXT,USTXT,CNKI,IEEE: 连续，持续，不间断，子任务，分配，调度，调配，分派，分发，指派，划分，拆，分割，分成，切割，分为，分成，亲和，优先，continue，task，allocate，schedule，split，cut，apart，affinity，prior

C. 相关文件

类 型*	引用文件，必要时，指明相关段落	相关的权利要求
Y	US 2021216557 A1 (EMC IP HOLDING CO LLC) 2021年7月15日 (2021-07-15) 说明书第18-96段	1-16
Y	CN 105260244 A (北京奇艺世纪科技有限公司) 2016年1月20日 (2016-01-20) 说明书第65-123段	1-16
Y	US 5761512 A (IBM) 1998年6月2日 (1998-06-02) 说明书第2-7栏	1-16
Y	US 2010153963 A1 (KAKARLAMUDI, Subbarao 等) 2010年6月17日 (2010-06-17) 说明书第46段	1-16
Y	WO 2022067531 A1 (深圳大学) 2022年4月7日 (2022-04-07) 说明书第6-11页	1-16
A	US 2018115496 A1 (ADVANCED MICRO DEVICES, INC.) 2018年4月26日 (2018-04-26) 全文	1-16
A	CN 114924858 A (中国银行股份有限公司) 2022年8月19日 (2022-08-19) 全文	1-16

☑ 其余文件在C栏的续页中列出。 ☑ 见同族专利附件。

* 引用文件的具体类型：
“A” 认为不特别相关的表示了现有技术一般状态的文件
“D” 申请人在国际申请中引证的文件
“E” 在国际申请日的当天或之后公布的在先申请或专利
“L” 可能对优先权要求构成怀疑的文件，或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件（如具体说明的）
“O” 涉及口头公开、使用、展览或其他方式公开的文件
“P” 公布日先于国际申请日但迟于所要求的优先权日的文件
“T” 在申请日或优先权日之后公布，与申请不相抵触，但为了理解发明之理论或原理的在后文件
“X” 特别相关的文件，单独考虑该文件，认定要求保护的发明不是新颖的或不具有创造性
“Y” 特别相关的文件，当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时，要求保护的发明不具有创造性
“&” 同族专利的文件

国际检索实际完成的日期	国际检索报告邮寄日期
2023年8月14日	2023年8月22日
ISA/CN的名称和邮寄地址 中国国家知识产权局 中国北京市海淀区蓟门桥西土城路6号 100088	受权官员 武晓冬 电话号码 (+86) 010-53961385

C. 相关文件

类 型*	引用文件，必要时，指明相关段落	相关的权利要求
A	US 2009300623 A1 (BANSAL, Nikhil 等) 2009年12月3日 (2009-12-03) 全文	1-16
A	CN 114741207 A (之江实验室) 2022年7月12日 (2022-07-12) 全文	1-16

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2023/104517

检索报告引用的专利文件			公布日 (年/月/日)	同族专利			公布日 (年/月/日)
US	2021216557	A1	2021年7月15日	US	11436231	B2	2022年9月6日
CN	105260244	A	2016年1月20日		无		
US	5761512	A	1998年6月2日	GB	9626242	D0	1997年2月5日
				GB	2308906	A	1997年7月9日
				GB	2308906	B	1999年10月20日
US	2010153963	A1	2010年6月17日	US	9390130	B2	2016年7月12日
WO	2022067531	A1	2022年4月7日	CN	112214319	A	2021年1月12日
US	2018115496	A1	2018年4月26日	JP	2019537104	A	2019年12月19日
				EP	3529697	A1	2019年8月28日
				KR	20190070915	A	2019年6月21日
				WO	2018075131	A1	2018年4月26日
				CN	109791507	A	2019年5月21日
				IN	201917006051	A	2019年4月26日
CN	114924858	A	2022年8月19日		无		
US	2009300623	A1	2009年12月3日	US	8458720	B2	2013年6月4日
CN	114741207	A	2022年7月12日		无		