(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
2016 年 12 月 8 日 (08.12.2016)

WIPO | PCT

(10) 国际公布号
WO 2016/192604 A1

(51) 国际专利分类号:
G06F 9/50 (2006.01)

(21) 国际申请号: PCT/CN2016/083889

(22) 国际申请日: 2016 年 5 月 30 日 (30.05.2016)

(25) 申请语言: 中文

(26) 公布语言: 中文

(30) 优先权:
201510303336.X 2015 年 6 月 5 日 (05.06.2015) CN

(71) 申请人: 阿里巴巴集团控股有限公司 (ALIBABA GROUP HOLDING LIMITED) [—/CN]; 英属开曼群岛大开曼资本大厦一座四层 847 号邮箱, Cayman Islands (KY)。

(72) 发明人: 吉悦 (JI, Yue); 中国浙江省杭州市余杭区文一西路 969 号 3 号楼 5 楼阿里巴巴集团法务部, Zhejiang 311121 (CN)。

(74) 代理人: 北京三友知识产权代理有限公司 (BEIJING SANYOU INTELLECTUAL PROPERTY AGENCY LTD.); 中国北京市金融街 35 号国际企业大厦 A 座 16 层, Beijing 100033 (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, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, 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, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, 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, GH, GM, KE, LR, LS, MW, MZ, NA, RW, 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, 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)。

[见续页]

(54) Title: VISUALIZATION METHOD, DEVICE AND SYSTEM FOR GLOBAL TASK NODE DEPENDENCE RELATIONSHIP

(54) 发明名称 : 一种全局任务节点依赖关系可视化方法、装置和系统

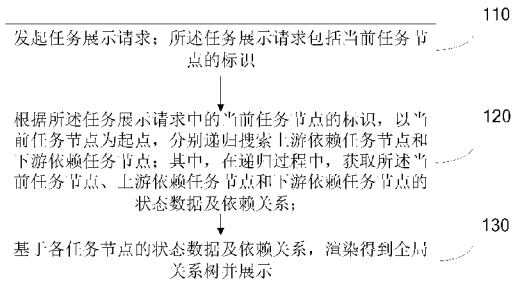

图 1

110 INITIATING A TASK PRESENTATION REQUEST, THE TASK PRESENTATION REQUEST COMPRISING AN IDENTIFIER OF A CURRENT TASK NODE

120 ACCORDING TO THE IDENTIFIER OF THE CURRENT TASK NODE IN THE TASK PRESENTATION REQUEST AND TAKING THE CURRENT TASK NODE AS A STARTING POINT, RECURSIVELY SEARCHING AN UPSTREAM DEPENDENT TASK NODE AND A DOWNSTREAM DEPENDENT TASK NODE RESPECTIVELY, WHEREIN, IN THE RECURSION PROCESS, STATE DATA AND DEPENDENCE RELATIONSHIPS OF THE CURRENT TASK NODE, THE UPSTREAM DEPENDENT TASK NODE AND THE DOWNSTREAM DEPENDENT TASK NODE ARE ACQUIRED

130 ON THE BASIS OF THE STATE DATA AND THE DEPENDENCE RELATIONSHIPS OF THE TASK NODES, OBTAINING A GLOBAL RELATIONSHIP TREE THROUGH RENDERING AND PRESENTING THE GLOBAL RELATIONSHIP TREE

(57) Abstract: A visualization method, device and system for a global task node dependence relationship, which relates to the technical field of computers. The method comprises: initiating a task presentation request, the task presentation request comprising an identifier of a current task node (110); according to the identifier of the current task node in the task presentation request and taking the current task node as a starting point, recursively searching an upstream dependent task node and a downstream dependent task node respectively, wherein, in the recursion process, state data and a dependence relationship of the current task node, the upstream dependent task node and the downstream dependent task node are acquired (120); and on the basis of the state data and the dependence relationship of the task nodes, obtaining a global relationship tree through rendering and presenting the global relationship tree (130). The method saves on user operating steps, and due to a comprehensive dependence relationship of a current stage, the method can present upstream and downstream dependence relationships of the current task node at the same time, which facilitates the unified management by the user of tasks deployed thereby.

(57) 摘要:

[见续页]

WO 2016/192604 A1

本国际公布:

— 包括国际检索报告(条约第 21 条(3))。

一种全局任务节点依赖关系可视化方法、装置和系统，涉及计算机技术领域。所述方法包括：发起任务展示请求；所述任务展示请求包括当前任务节点的标识(110)；根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖任务节点和下游依赖任务节点；其中，在递归过程中，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系(120)；基于各任务节点的状态数据及依赖关系，渲染得到全局关系树并展示(130)。本方法节省了用户的操作步骤，并且由于当前阶段的依赖关系全面，可同时展示当前任务节点的上、下游的依赖关系，方便用户统一管理其部署的任务。

一种全局任务节点依赖关系可视化方法、装置和系统

本申请要求 2015 年 06 月 05 日递交的申请号为 201510303336.X、发明名称为“一种全局任务节点依赖关系可视化方法、装置和系统”的中国专利申请的优先权，其全部内容通过引用结合在本申请中。

技术领域

本申请涉及计算机技术领域，特别是涉及一种全局任务节点依赖关系可视化方法、一种全局任务节点依赖关系可视化装置和一种全局任务节点依赖关系可视化系统。

背景技术

随着大数据的应用，很多数据处理平台会对各种数据进行处理，比如 ODPS 平台（Open Data Processing Service，开放数据处理服务），该数据平台提供针对 TB/PB 级数据、实时性要求不高的分布式处理能力，可应用于数据分析、挖掘、商业智能等领域。

那么在数据开发过程中，针对某个数据业务，开发者会将其拆解为一系列的任务部署到数据平台中，该任务是运行在数据平台上的最小调度作业单元。其中，各个任务之间可能存在某种依赖关系，每个任务可以认为是该数据业务中的一个任务节点。而对于数据开发者，其需要对自己部署的任务节点进行监控。

在先的监控任务节点监控方法，其是用户在客户端发送 http 请求，将任务节点 ID 传输到后台服务器，后台服务器根据该 ID 拿到下一级的任务节点，然后将该级任务节点返回给客户端，客户端则展现该级任务节点。但是上述过程，只能展示当前任务节点的下一级任务节点，展示片面；并且，对于一个任务节点 A，需要用户逐级的多次点击操作才能得到该任务节点的部分的下游依赖的任务节点，从而得到下游依赖关系，比如任务节点 A 下一级的任务节点包括 A1、A2、A3，用户需要分别点击 A1、A2、A3 才能得到 A1、A2、A3 各自的下一级任务节点，从而得到任务节点 A 的部分下游依赖关系，因此，上述过程用户操作繁琐，其得到依赖关系也只是下游依赖关系，不够全面。

发明内容

鉴于上述问题，提出了本申请实施例以便提供一种克服上述问题或者至少部分地解决上述问题的一种全局任务节点依赖关系可视化方法和相应的一种全局任务节点依赖关系可视化装置，以及一种全局任务节点依赖关系可视化系统。

为了解决上述问题，本申请公开了一种全局任务节点依赖关系可视化方法，包括：

发起任务展示请求；所述任务展示请求包括当前任务节点的标识；

根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖任务节点和下游依赖任务节点；其中，在递归过程中，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系；

基于各任务节点的状态数据及依赖关系，渲染得到全局关系树并展示。

本申请还公开了一种全局任务节点依赖关系可视化装置，包括：

展示请求发起模块，用于发起任务展示请求；所述任务展示请求包括当前任务节点的标识；

递归搜索模块，用于根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖任务节点和下游依赖任务节点；其中，在递归过程中，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系；

渲染展示模块，用于基于各任务节点的状态数据及依赖关系，渲染得到全局关系树并展示。

本申请实施例包括以下优点：

本申请实施例对于用户当前点击的一个任务节点，发起一次包括当前任务节点的标识的任务展示请求，然后可根据当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖任务节点和下游依赖任务节点，并且可在递归过程中获取当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系，那么即可基于当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系，渲染得到全局关系树并展示。上述过程中，本申请实施例只需用户一次点击操作，即只需客户端发送一次任务展示请求，即可将当前任务节点的上游依赖节点和下游依赖节点一起渲染为全局关系树展示给用户，节省了用户的操作步骤，并且由于当前阶段的依赖关系全面，可同时展示当前任务节点的上、下游的依赖关系，方便用户统一管理其部署的任务。

附图说明

图 1 是本申请的一种全局任务节点依赖关系可视化方法实施例的步骤流程图；

图 1A 是任务节点之间的依赖关系的一种示例；

图 1B 是针对图 1A 中任务节点 6 的下游依赖任务节点的正向渲染示例；

图 1C 针对图 1A 中任务节点 6 的上游依赖任务节点的正向渲染示例；

图 1D 是针对图 1B 和图 1C 中的渲染结果进行拼接后渲染的示例；

图 2 是本申请另一种全局任务节点依赖关系可视化方法实施例的步骤流程图；

图 3 是本申请的一种全局任务节点依赖关系可视化装置实施例的结构框图；

图 4 是本申请的另一种全局任务节点依赖关系可视化装置实施例的结构框图；

图 5 是本申请的一种全局任务节点依赖关系可视化系统实施例的结构框图。

具体实施方式

为使本申请的上述目的、特征和优点能够更加明显易懂，下面结合附图和具体实施方式对本申请作进一步详细的说明。

本申请实施例的核心构思之一在于，通过用户针对一任务节点所触发的一次性任务展示请求到后台服务器，后台服务器递归搜索该任务节点的上游任务节点和下游任务节点，然后将递归过程中获取到的各任务节点的状态数据及依赖关系返回给前端，在前端对各任务节点的状态数据以依赖关系进行渲染。从而使用户一次性操作就可以对某个任务节点的全局任务依赖关系进行渲染展示，大大降低用户的操作次数，方便用户对全局的任务节点进行统一管理，提升了产品的易用性。

实施例一

参照图 1，示出了本申请的一种全局任务节点依赖关系可视化方法实施例的步骤流程图，具体可以包括如下步骤：

步骤 110，发起任务展示请求；所述任务展示请求包括当前任务节点的标识；

在本申请实施例中，以 ODPS 平台为例，在对某个数据业务进行部署时，会将数据业务拆分为一系列任务部署到 ODPS 平台中，上述任务为运行在数据平台上的最小调度作业单元。而由于该一系列任务的各个任务之间存在依赖关系，每个任务可以看做一个任务节点，每个任务节点可能与一个或多个任务节点存在依赖关系。如图 1A 的示例所示，任务节点 1、2、3 处理的结果向下传输到任务节点 5，任务节点 5 处理的结果向下传输到任务节点 7，任务节点 7 处理的结果向下传输到任务节点 10、11，任务节点 0 处理的数据向下传输到任务节点 3，任务节点 3、4 处理的结果向下传输到任务节点 6，任务节点 6 处理的结果向下传输到任务节点 7、8、9，任务节点 8 处理的结果向下传输到任务节点 12，其他情况以此类推。

那么如果用户想要查看任务节点 6 的全局依赖关系，则可以在客户端针对任务节点

6 发起一次任务展示请求，该任务展示请求包括了任务节点 6 的 ID，比如“6”。

在具体实现中，本申请实施例对于任务展示请求，可以采用 ajax 请求的形式到后台服务器。ajax（Asynchronous JavaScript And XML，异步的 JavaScript 和 XML），其中 JavaScript（脚本语言），XML（可扩展标记语言，Extensible Markup Language）。Ajax 可以在不刷新页面的情况下对获取服务器的数据然后实时更新到页面中。Ajax 的核心是 JavaScript 对象 XmlHttpRequest。该对象在 Internet Explorer 5 中首次引入，它是一种支持异步请求的技术。本申请实施例中，可以将当前任务节点标识构造 ajax 请求，其过程大致如下：

1）创建 XMLHttpRequest 对象实例。

2）利用 XMLHttpRequest 对象的 onreadystatechange 属性，设定 XMLHttpRequest 对象的回调函数，该回调函数是用来处理服务器响应信息的。

3）设定请求属性：设定 HTTP（Hypertext transfer protocol，超文本传送协议）方法，利用 XMLHttpRequest 对象的 open()方法设定目标 URL。其中 HTTP 方法包括：GET 方法或 POST 方法，在此处可以选择一个，并且在该 HTTP 方法中写入当前任务节点的 ID。其中目标 URL 为后台服务器的地址。

如此即构建了一个 ajax 请求，然后将该 ajax 请求发送至服务器。在实际应用中可以利用 XMLHttpRequest 对象的 send()方法将 ajax 请求发送至后台服务器。

当然，本申请实施例实施例中，对于任务展示请求也可以采用其他形式的请求，本申请实施例不对其加以限定。

步骤 120，根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖任务节点和下游依赖任务节点；其中，在递归过程中，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系；

如前述例子，接收到任务展示请求后，解析该任务展示请求中的当前任务节点的标识，比如 6。那么如图 1A，可以 6 所在的任务节点为起点，分别向上游和下游递归搜索，搜索任务节点 6 的上游依赖节点和下游依赖节点。任务节点 6 的上游依赖为：任务节点 6 依赖任务节点 3 和任务节点 4，任务节点 3 依赖任务节点 0。任务节点 6 的下游依赖为：任务节点 6 依赖任务节点 8 和任务节点 9，任务节点 8 依赖任务节点 12。

在本申请实施例的递归搜索的过程中，每递归到一个任务节点，则获取该任务节点的状态数据及依赖关系。

其中，任务节点的状态数据包括：正在运行、运行成功、运行失败。

当然，在本申请实施例中，还可获取每个任务节点的其他数据，比如任务节点的名称，任务节点是否为叶子节点，任务节点相对任务展示请求中的当前任务节点的级别深度等。

在本申请实施例中，在后台服务器中预先定义了任务节点关系数据表，其至少包括四个字段：任务节点 ID、节点数据表 ID、依赖类型、节点数据表类型，具体如表（一）

action_id	table_id	type	table_type_id
6	001	2	2

表（一）

其中，action_id 为任务节点 ID。

table_id 为节点数据表 ID。

type 为依赖类型，指明节点数据表是被该任务节点上游依赖还是下游依赖，比如 1 为上游依赖，2 为下游依赖。每个节点数据表中记录了该任务节点 ID 依赖的一级任务节点。对于表（一）中的任务节点 6，其对应的节点数据表为 001，任务节点 6 向上游依赖节点数据表 001 中的一级任务节点，赖节点数据表 001 在包括任务节点{3、4}。当然，对于一个任务节点，也有向下游依赖的节点数据表。当然，如果某个任务节点为叶子节点，则其不再有下游依赖的节点数据表，对应该任务节点的 type 可设为 0，表示对于该条递归路径，到此结束，可以递归其他未被递归的路径。

table_type_id 节点数据表类型，如如 odps 表,ump 表，

那么本申请实施例则可以根据上述表分别进行上游递归搜索和下游递归搜索，可以根据表（一）的内容获取某个任务节点的依赖关系，然后可根据搜索到的 ID 去实际的运行该任务节点的计算节点中获取该任务节点的状态数据。当然，任务节点的状态数据也可以实时加到表（一）的相应任务节点的相应状态数据字段中，那么在本申请递归时，即可直接通过表（一）获取到某个任务节点的状态数据。

优选的，所述步骤 120，包括：

子步骤 A10，以当前任务节点为起点，分别采用深度优先搜索算法向上游、下游进行递归搜索，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系。

在本申请实施例中，以当前任务节点为起点向上游递归搜索和以当前任务节点为起点向下游递归搜索的方法可以采用 dfs 算法（Depth First Search，深度优先搜索算法）。

在深度优先搜索算法中，对于最新发现的节点 v，如果它还有以此为起点而未探测

到的边，就沿此边继续探测下去。当节点 v 的所有边都已被探寻过，搜索将回溯到还存在有没被探索的边节点进行搜索。上述过程一直进行到已发现从源结点可达的所有节点为止。如果还存在未被发现的节点，则选择其中一个作为源节点并重复以上过程，整个进程反复进行直到所有节点都被发现为止。

比如对于以任务节点 6 向上游递归，那么首先根据标识 6，递归搜索到任务节点 6，则获取任务节点 6 的状态数据，其向上依赖的任务节点为 3、4，3、4 为第 1 级上游依赖；再向上递归搜索到任务节点 3，则获取任务节点 3 的状态数据，其向上依赖任务节点 0，0 为第 2 级上游依赖；再向上递归搜索到任务节点 0，则获取任务节点 0 的状态数据，由于 0 已经为第 2 级上游依赖，则不再递归搜索，该条递归路径结束；再向上递归搜索到任务节点 4，则获取任务节点 4 的状态数据，任务节点 4 向上依赖关系为空，没有第 2 级上游依赖，则该条递归路径结束。至此，任务节点 6 的 2 级之内上游依赖节点递归搜索完毕，也获取到了各个上游依赖节点的状态数据和依赖关系。

比如对于以任务节点 6 向上游递归，则首先根据标识 6，递归搜索任务节点 6，获取 6 的状态数据，其向下依赖的任务节点为 7、8、9；再向下递归搜索到任务节点 7，则获取任务节点 7 的状态数据，其向下依赖任务节点 10、11，再向下递归搜索到任务节点 10，则获取任务节点 10 的状态数据，任务节点 10 向下的依赖关系为空，该条递归路径结束；再递归搜索到 11，则获取任务节点 11 的状态数据，任务节点 11 向下的依赖关系为空，该条递归路径结束；再递归搜索到任务节点 8，则获取任务节点 8 的状态数据，任务节点 8 向上依赖关系为 12，则获取任务节点 12 的状态数据，任务节点 12 向下的依赖关系为空，则该条递归路径结束；再向下递归搜索到任务节点 9，则获取任务节点 9 的状态数据，任务节点 9 向下的依赖关系为空，则该条递归路径结束。至此，任务节点 6 的所有下游依赖节点递归搜索完毕，也获取到了各个上游依赖节点的状态数据和依赖关系。

当然，本申请实施例中也可以采用其他递归方法，在此不对其加以限制。

优选的，在步骤 120 之后，还包括：

步骤 122，采用预置数据结构，将所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系存储为数据文件；所述预置数据结构包括：第一字段：用于存储递归形式的上游依赖关系；第二字段：用于存储递归形式的下游依赖关系；第三字段：用于存储当前递归级的任务节点标识；第四字段：用于存储当前递归级的任务节点的状态数据。

在本申请实施例中，可预先设置一预置数据结构，该预置数据结构包括：第一字段：用于存储递归形式的上游依赖关系；第二字段：用于存储递归形式的下游依赖关系；第三字段：用于存储当前递归级的任务节点标识；第四字段：用于存储当前递归级的任务节点的状态数据。

那么可以通过该预置数据结构将所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系存储为数据文件。

以包括前述第一字段、第二字段、第三字段、第四字段的预置数据结构为例，其示例如下：

{“data”：{“deptreenodelist” [

Objiect{…}

]

{“children” [

Objiect{…}

]

“actionid”:123456，

“ status”:1，

“program”：“”

}

“errMsg”：“”

}

其中，deptreenodelist 为第一字段，在其中的 Objiect{…}中存储递归形式的上游依赖关系；children 第二字段，在其中的 Objiect{…}中存储递归形式的下游依赖关系；actionid 为第三字段，用于存储当前递归级的任务节点标识；status 为第四字段，用于存储当前递归级的任务节点的状态数据，比如 1 表示运行成功，2 表示运行失败，3 表示正在运行；program 表示任务节点对应的源程序，也就是任务实际执行的代码；errMsg 表示给客户端返回的错误标识，例如网络不通，数据库错误等系统级错误。

以前述的任务节点 6 的上游依赖节点为例 ，其存储的结果如下：

{“data”：{“deptreenodelist” [

Objiect{

{“data”：{“deptreenodelist” [

Objiect{
{“data”：{“deptreenodelist” [
Objiect{null}
]
{“children” [
Objiect{null}
]
“actionid”:0，
“ status”:1，
“program”：“”
}
“errMsg”：“”
}
}
]
{“children” [
Objiect{null}
]
“actionid”:3，
“ status”:1，
“program”：“”
}
“errMsg”：“”
}
{“data”：{“deptreenodelist” [
Objiect{null}
]
{“children” [
Objiect{null}
]
“actionid”:4，
“ status”:1，

“program”：“”

}

“errMsg”：“”

}

}

]

{“children” [

Objiect{…}

]

“actionid”:6，

“ status”:1，

“program”：“”

}

“errMsg”：“”

}

上述示例中以预置的数据结构为基础，对每个任务节点的状态数据和依赖关系进行嵌套存储获得递归形式的上游依赖关系，每个任务节点的状态数据在相应层级的数据结构里存储。最外层是任务节点 6 的数据结构，然后在任务节点 6 的数据结构的第一字段中平行嵌入了下一级任务节点的数据结构，即任务节点 3、4 的数据结构，然后在任务节点 3 的数据结构的第一字段中嵌入了任务节点 0 的数据结构。

同理，可以在最外层的数据结构的 children 字段中嵌套存储下游依赖任务节点，获得递归形式的下游依赖关系。

当然，本申请实施例中，如果对于一个任务节点，还有其他的信息，比如任务节点的名称，任务节点是否为叶子节点，任务节点相对任务展示请求中的当前任务节点的级别深度等，可以在前述预置数据结构中添加相应字段。比如：

{“data”：{“deptreenodelist” [

Objiect{…}

]

{“children” [

Objiect{…}

]

“actionid”:6，

“is_leaf”：-1，

“depth”：0，

“name”：“qingxiao”，

“ status”:3，

“current”：，

“program”：“”

}

“errMsg”：“”

}

其中，is_leaf 表示任务节点是否为叶子节点，可设置-1 为不是，1 为是；depth 表示任务节点相对任务展示请求中的当前任务节点的级别深度，比如对于任务节点 6 来说，其级别为 0，每增加一级，数字加 1；name 的名称为表示任务节点的名称，比如任务节点 6 的名称 qingxiao。

那么，通过上述预置的数据结构，则可以将当前任务节点、上游依赖任务节点、下游依赖节点的各种信息和依赖关系通过数据结构嵌套的方式，实现递归形式的上游依赖关系的存储和递归形式的下游依赖关系的存储，从而获得一个数据文件。

当然，本申请实施例中，还可以每递归搜索到一个任务节点，则将该任务节点的状态数据和依赖关系按照上述预置数据结构存储。

优选的，所述步骤 120 包括：

步骤 A20，根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，在分布式缓存中分别递归搜索上游依赖任务节点和下游依赖任务节点；

步骤 A22，当在分布式缓存中未能获取到上游依赖任务节点和下游依赖任务节点，则进入数据库中分别递归搜索上游依赖任务节点和下游依赖任务节点。

在本申请实施例中，各任务节点的数据，比如前述表（一）的数据，预先是存储在后台服务器的数据库中，而当后台服务器接收到客户端发送的任务展示请求后，如果直接从后台服务器的数据库中进行递归搜索，则处理的时间长，客户端的等待时间也较长，那么为了减少客户端的等待时间，本申请实施例后台服务器可先将部分或者全部任务节点的数据以及依赖关系放入分布式缓存中。

那么在根据所述任务展示请求中的当前任务节点 ID，首先在分布式缓存中进行上游

递归搜索获取上游依赖任务节点，并进行下游递归搜索获取下游依赖任务节点。如果在分布式缓存中没有递归搜索到上游依赖任务节点和或下游依赖任务节点，则可直接进入步骤 B12。

步骤 130，基于各任务节点的状态数据及依赖关系，渲染得到全局关系树并展示。

在本申请实施例对于当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系,可以在客户端进行渲染,将其渲染为全局关系树进行展示。

在本申请实施例中，对于每个任务节点的图标，可以将其状态数据以颜色的形式进行展示，如图 1D 中，白色表示运行成功、比如任务节点 10 的颜色表示正在运行，任务 12 的颜色表示运行失败。当然也可以通过其他方式进行展示。

当然，也可以将任务节点的其他信息，比如 ID 渲染到当前节点的图标中，如图 1D。对于任务节点的详细信息，可以在鼠标滑动到相应任务节点的渲染图上时，以弹出框进行展示。当然任务节点的详细信息也可以直接渲染到该任务节点的图标中。

具体的对任务节点的图标的渲染方式，本申请不对其加以限制。

优选的，在前述优选的步骤 122 的基础之上，所述步骤 130 包括：

子步骤 132，基于所述数据文件中，以预置数据结构存储的当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系，渲染得到全局关系树并展示。

后台服务器在采用预置数据结构将所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系存储的数据文件后，可将该数据文件返回至客户端，那么客户端可按照预先的约定解析该数据文件，得到该数据文件中当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系，其中如前述例子中：第一字段存储了递归形式的上游依赖关系；第二字段存储了递归形式的下游依赖关系；第三字段存储了当前任务节点标识；第四字段存储了任务节点的状态数据；如果还有其他字段，则按照相应的预定提取相应数据。

然后客户端可基于上述预置数据结构中的数据进行渲染，得到全局关系树进行展示。在本发明实施例中客户端可以为浏览器，优选的可以为支持 html5 canvas 的浏览器，该浏览器可以流畅的支持 ajax 功能。

优选的，步骤 130 包括：

子步骤 A30，以当前任务节点为根节点，根据当前任务节点和下游依赖任务节点的状态数据及依赖关系，进行正向渲染获得第一多叉树；

在本申请实施例中，可以采用数据化可视工具 d3.js(bsd liences)进行渲染， d3.js 是

一种数据可视化框架，其开源协议为 bsd 协议。那么本申请实施例可采用 d3.js 中的 treelayout，以当前任务节点为根节点，对当前任何节点和下游任务节点进行正向渲染，得到一个正向的多叉树。

子步骤 A32，以指定的任务节点为根节点，根据当前任务节点和上游依赖任务节点的状态数据及依赖关系，进行反向渲染获得第二多叉树；

本申请实施例可采用 d3.js 中的 treelayout，以当前任务节点为根节点，对当前任何节点和上游任务节点进行反向渲染，得到一个反向的多叉树。当然，本发明实施例对 treelayout 的功能进行了扩展，使其可以进行反向渲染。

优选的，所述子步骤 A32 包括：

子步骤 A321，以指定的任务节点为根节点，根据当前任务节点和下游依赖任务节点的状态数据及依赖关系，进行正向渲染获得与第一多叉树同向的第二多叉树；

在实际应用中，d3.js 的 treelayout 只支持对 children 字段的节点进行渲染，并且只能渲染为一个多叉树，在本申请实施例中 children 性质的节点为下游依赖任务节点。而为了实现本申请对当前任务节点的上游任务依赖节点的渲染，本申请实施例在 treelayout 中复制一套 children 的渲染方式，并将复制的渲染方式中的 children 字段替换为 deptreenodelist，如此，本申请实施例对上游依赖任务节点的渲染过程实质上与对下游依赖任务节点的渲染过程一致。得到两个同向的多叉树。

那么对于前述以预置数据结构存储的数据文件，前端接收到该数据文件之后，解析得到前述预置数据结构，可以分别以当前任务节点为起点，分别对 children 和 deptreenodelist 字段下的任务节点进行渲染，得到两个同向的多叉树。如图 1B，为渲染得到下游依赖任务节点的第一多叉树示例；如图 1C，为渲染得到上游依赖任务节点的第一多叉树示例。

子步骤 A322，将与第一多叉树同向的第二多叉树进行倒置，获得最终的第二多叉树。

在得到同向的第一多叉树和第二多叉树之后，为了得到针对当前任务节点全局关系树，则需要将第二多叉树进行倒置，使其与第一多叉树方向相反。

当然，本申请实施例中，对于第一多叉树和第二多叉树，可以任意改变其方向，只需要保证两者的多叉树互相不影响依赖关系的展现即可，比如第一多叉树向下展示，第二多叉树向上展示，本申请实施例不对其加以限制。

子步骤 A34，将第一多叉树和第二多叉树拼接为全局关系树并展示。

本申请实施例可将第一多叉树和第二多叉树进行拼接，上述拼接可将两个多叉树的

当前任务节点的渲染图重叠，即可得到当前任务节点的全局关系树，然后即可将该全局关系树展示在客户端。如图 1D，为前述例子的展示结果。

在数据处理平台的数据开发过程中，在在先的技术中，技术人员惯性的会设置对下游的任务节点进行展示，并且根据点击操作，获取每个节点的下一级任务节点的数据进行展示。而本申请实施例对于用户当前点击的一个任务节点，发起一次包括当前任务节点的标识的任务展示请求，然后可根据当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖任务节点和下游依赖任务节点，并且可在递归过程中获取当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系，那么即可基于当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系，渲染得到全局关系树并展示。

上述过程中，本申请实施例只需用户一次点击操作，即只需客户端发送一次任务展示请求，即可将当前任务节点的上游依赖节点和下游依赖节点一起渲染为全局关系树展示给用户，节省了用户的操作步骤，并且由于当前阶段的依赖关系全面，可同时展示当前任务节点的上、下游的依赖关系，方便用户统一管理其部署的任务。

实施例二

参照图 2，示出了本申请的一种全局任务节点依赖关系可视化方法实施例的步骤流程图，具体可以包括如下步骤：

步骤 210，发起任务展示请求；所述任务展示请求包括当前任务节点的标识，和以所述当前任务节点为起点的上游依赖层级和下游依赖层级；

在本发明实施例中，用户可以在客户端设置需要展示的上游依赖层级和下游依赖层级，同时指定当前任务节点的 ID，然后通过任务展示请求发送到后台服务器。其中上游依赖层级和下游依赖层级均为相对当前任务节点的层级。比如图 1A 中的任务节点 6，直接与任务节点 6 关联的任务节点 3、4 为相对任务节点 6 的第 1 级上游依赖，直接与任务节点 3 关联的 0 为相对任务节点 6 第 2 级上游依赖；直接与任务节点 6 关联的任务节点 7、8、9 为相对任务节点 6 的第 1 级下游依赖，直接与任务节点 7 关联的 10、11 为相对任务节点 6 第 2 级上游依赖，接与任务节点 8 关联的 12 为相对任务节点 6 第 2 级上游依赖。其他情况以此类推。

步骤 220，根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖层级之内的上游依赖任务节点和下游依赖层级之内的下游依

赖任务节点；其中，在递归过程中，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系；

后台服务器收到前述任务展示请求之后，则根据当前任务节点的标识，可向上游递归搜索， 比如规定上游依赖层级为 2，下游依赖层级为 2，则

对于以任务节点 6 向上游递归，那么首先根据标识 6，递归搜索到任务节点 6，则获取任务节点 6 的状态数据，其向上依赖的任务节点为 3、4；再向上递归搜索到任务节点 3，则获取任务节点 3 的状态数据，其向上依赖任务节点 0；再向上递归搜索到任务节点 0，则获取任务节点 0 的状态数据，任务节点 0 向上的依赖关系为空，该条递归路径结束；再向上递归搜索到任务节点 4，则获取任务节点 4 的状态数据，任务节点 4 向上依赖关系为空，则该条递归路径结束。至此，任务节点 6 的所有上游依赖节点递归搜索完毕，也获取到了各个上游依赖节点的状态数据和依赖关系。

比如对于以任务节点 6 向上游递归，则首先根据标识 6，递归搜索任务节点 6，获取 6 的状态数据，其向下依赖的任务节点为 7、8、9，7、8、9 为第 1 级下游依赖；再向下递归搜索到任务节点 7，则获取任务节点 7 的状态数据，其向下依赖任务节点 10、11，10、11 为第 2 级下游依赖，再向下递归搜索到任务节点 10，则获取任务节点 10 的状态数据，由于已经是第 2 级下游依赖，则不再向下递归，即使下游还有任务节点，因此该条递归路径结束；再递归搜索到 11，则获取任务节点 11 的状态数据，由于已经是第 2 级下游依赖，则不再向下递归，即使下游还有任务节点，因此该条递归路径结束；再递归搜索到任务节点 8，则获取任务节点 8 的状态数据，任务节点 8 向上依赖关系为 12，则获取任务节点 12 的状态数据，由于已经是第 2 级下游依赖，则不再向下递归，即使下游还有任务节点，因此该条递归路径结束；再向下递归搜索到任务节点 9，则获取任务节点 9 的状态数据，任务节点 9 向下的依赖关系为空，则该条递归路径结束。至此，任务节点 6 的 2 级之内的下游依赖节点递归搜索完毕，也获取到了各个上游依赖节点的状态数据和依赖关系。

当然，在后台服务器中，也可在数据库中预置实施例一中的表（一），递归搜索时根据表（一）进行递归。并且表（一）中的内容可以部分或者全部的放置于分布式缓存中。

当然，在后台服务器中，也可将获取到的当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系采用类似实施例 1 中的采用预置数据结构，存储为数据文件，然后将该数据文件返回客户端，客户端可解析该数据文件进行渲染，得到全

局关系树。

步骤230，基于各任务节点的状态数据及依赖关系，渲染得到全局关系树并展示。

本申请实施例对于用户当前点击的一个任务节点，发起一次包括当前任务节点的标识的任务展示请求和上游依赖层级和下游依赖层级，然后可根据当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖层级之内的上游依赖任务节点和下游依赖层级之内的下游依赖任务节点，并且可在递归过程中获取当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系，那么即可基于当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系，渲染得到全局关系树并展示。

上述过程中，本申请实施例只需用户一次点击操作，即只需客户端发送一次任务展示请求，即可将当前任务节点的上游依赖节点和下游依赖节点一起渲染为全局关系树展示给用户，节省了用户的操作步骤，并且由于当前阶段的依赖关系全面，可同时展示当前任务节点的上、下游的依赖关系，并且由于可以设置上、下游依赖层级，进一步方便用户统一管理其部署的任务。

需要说明的是，对于方法实施例，为了简单描述，故将其都表述为一系列的动作组合，但是本领域技术人员应该知悉，本申请实施例并不受所描述的动作顺序的限制，因为依据本申请实施例，某些步骤可以采用其他顺序或者同时进行。其次，本领域技术人员也应该知悉，说明书中所描述的实施例均属于优选实施例，所涉及的动作并不一定是本申请实施例所必须的。

实施例三

参照图 3，示出了本申请的一种全局任务节点关系可视化装置实施例的结构框图，具体可以包括如下模块：

展示请求发起模块 310，用于发起任务展示请求；所述任务展示请求包括当前任务节点的标识；

递归搜索模块 320，用于根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖任务节点和下游依赖任务节点；其中，在递归过程中，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系；

渲染展示模块 330，用于基于各任务节点的状态数据及依赖关系，渲染得到全局关系树并展示。

优选的，在递归搜索模块320之后，还包括：

节点数据存储模块，用于采用预置数据结构，将所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系存储为数据文件；所述预置数据结构包括：第一字段：用于存储递归形式的上游依赖关系；第二字段：用于存储递归形式的下游依赖关系；第三字段：用于存储当前递归级的任务节点标识；第四字段：用于存储当前递归级的任务节点的状态数据；

进一步的，所述渲染展示模块包括：

第一渲染展示模块，用于基于所述数据文件中，以预置数据结构存储的当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系，渲染得到全局关系树并展示。

优选的，所述渲染展示模块 330 包括

第一多叉树渲染模块，用于以当前任务节点为根节点，根据当前任务节点和下游依赖任务节点的状态数据及依赖关系，进行正向渲染获得第一多叉树；

第二多叉树渲染模块，用于以指定的任务节点为根节点，根据当前任务节点和上游依赖任务节点的状态数据及依赖关系，进行反向渲染获得第二多叉树；

拼接展示模块，用于将第一多叉树和第二多叉树拼接为全局关系树并展示。

优选的，所述第二多叉树渲染模块包括：

同向渲染模块，用于以指定的任务节点为根节点，根据当前任务节点和下游依赖任务节点的状态数据及依赖关系，进行正向渲染获得与第一多叉树同向的第二多叉树；

倒置模块，用于将与第一多叉树同向的第二多叉树进行倒置，获得最终的第二多叉树。

优选的，所述递归搜索模块 320 包括：

深度递归搜索模块，用于以当前任务节点为起点，分别采用深度优先搜索算法向上游、下游进行递归搜索，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系。

优选的，所述递归搜索模块 320 包括：

分布式搜索模块，用于根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，在分布式缓存中分别递归搜索上游依赖任务节点和下游依赖任务节点；

数据库搜索模块，用于当在分布式缓存中未能获取到上游依赖任务节点和下游依赖任务节点，则进入数据库中分别递归搜索上游依赖任务节点和下游依赖任务节点。

实施例四

参照图 4，示出了本申请的一种全局任务节点关系可视化装置实施例的结构框图，具体可以包括如下模块：

展示请求发起模块 410，用于发起任务展示请求；所述任务展示请求包括当前任务节点的标识，和以所述当前任务节点为起点的上游依赖层级和下游依赖层级；

递归搜索模块 420，具体包括：

层级递归搜索模块 422，用于根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖层级之内的上游依赖任务节点和下游依赖层级之内的下游依赖任务节点；其中，在递归过程中，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系；

渲染展示模块 430，用于基于各任务节点的状态数据及依赖关系，渲染得到全局关系树并展示。

实施例五

参照图 5，示出了本申请的一种全局任务节点关系可视化系统实施例的结构框图，具体可以包括如下模块：

客户端 510 和后台服务器 520；

所述客户端 510 包括：

展示请求发起模块 512，用于发起任务展示请求；所述任务展示请求包括当前任务节点的标识；

渲染展示模块 514，用于基于各任务节点的状态数据及依赖关系，渲染得到全局关系树并展示；

所述后台服务器 520 包括：

递归搜索模块 522，用于根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖任务节点和下游依赖任务节点；其中，在递归过程中，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系，并将各任务节点的状态数据及依赖关系返回客户端。

优选的，所述后台服务器还包括：

节点数据存储模块，用于采用预置数据结构，将所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系存储为数据文件，并将所述数据文件返回客户端；所述预置数据结构包括：第一字段：用于存储递归形式的上游依赖关系；第二字段：用于存储递归形式的下游依赖关系；第三字段：用于存储当前递归级的任务节

点标识；第四字段：用于存储当前递归级的任务节点的状态数据；

进一步的，在所述客户端中，所述渲染模块包括：

第一渲染模块，用于基于所述数据文件中，以预置数据结构存储的当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系，渲染得到全局关系树并展示。

对于装置实施例而言，由于其与方法实施例基本相似，所以描述的比较简单，相关之处参见方法实施例的部分说明即可。

本说明书中的各个实施例均采用递进的方式描述，每个实施例重点说明的都是与其他实施例的不同之处，各个实施例之间相同相似的部分互相参见即可。

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

在一个典型的配置中，所述计算机设备包括一个或多个处理器 (CPU)、输入/输出接口、网络接口和内存。内存可能包括计算机可读介质中的非永久性存储器，随机存取存储器(RAM) 和/或非易失性内存等形式，如只读存储器(ROM)或闪存(flash RAM)。内存是计算机可读介质的示例。计算机可读介质包括永久性和非永久性、可移动和非可移动媒体可以由任何方法或技术来实现信息存储。信息可以是计算机可读指令、数据结构、程序的模块或其他数据。计算机的存储介质的例子包括，但不限于相变内存 (PRAM)、静态随机存取存储器 (SRAM)、动态随机存取存储器 (DRAM)、其他类型的随机存取存储器 (RAM)、只读存储器 (ROM)、电可擦除可编程只读存储器 (EEPROM)、快闪记忆体或其他内存技术、只读光盘只读存储器(CD-ROM)、数字多功能光盘(DVD)或其他光学存储、磁盒式磁带，磁带磁磁盘存储或其他磁性存储设备或任何其他非传输介质，可用于存储可以被计算设备访问的信息。按照本文中的界定，计算机可读介质不包括非持续性的电脑可读媒体(transitory media)，如调制的数据信号和载波。

本申请实施例是参照根据本申请实施例的方法、终端设备(系统)、和计算机程序产品的流程图和／或方框图来描述的。应理解可由计算机程序指令实现流程图和／或方框图中的每一流程和／或方框、以及流程图和／或方框图中的流程和／或方框的结合。可

提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理终端设备的处理器以产生一个机器，使得通过计算机或其他可编程数据处理终端设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和／或方框图一个方框或多个方框中指定的功能的装置。

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

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

尽管已描述了本申请实施例的优选实施例，但本领域内的技术人员一旦得知了基本创造性概念，则可对这些实施例做出另外的变更和修改。所以，所附权利要求意欲解释为包括优选实施例以及落入本申请实施例范围的所有变更和修改。

最后，还需要说明的是，在本文中，诸如第一和第二等之类的关系术语仅仅用来将一个实体或者操作与另一个实体或操作区分开来，而不一定要求或者暗示这些实体或操作之间存在任何这种实际的关系或者顺序。而且，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、物品或者终端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、物品或者终端设备所固有的要素。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、物品或者终端设备中还存在另外的相同要素。

以上对本申请所提供的一种全局任务节点关系可视化方法和一种全局任务节点关系可视化装置，进行了详细介绍，本文中应用了具体个例对本申请的原理及实施方式进行了阐述，以上实施例的说明只是用于帮助理解本申请的方法及其核心思想；同时，对于本领域的一般技术人员，依据本申请的思想，在具体实施方式及应用范围上均会有改变之处，综上所述，本说明书内容不应理解为对本申请的限制。

权 利 要 求 书

1、一种全局任务节点依赖关系可视化方法，其特征在于，包括：

发起任务展示请求；所述任务展示请求包括当前任务节点的标识；

根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖任务节点和下游依赖任务节点；其中，在递归过程中，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系；

基于各任务节点的状态数据及依赖关系，渲染得到全局关系树并展示。

2、根据权利要求 1 所述的方法，其特征在于，在获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系的步骤之后，还包括：

采用预置数据结构，将所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系存储为数据文件；所述预置数据结构包括：第一字段：用于存储递归形式的上游依赖关系；第二字段：用于存储递归形式的下游依赖关系；第三字段：用于存储当前递归级的任务节点标识；第四字段：用于存储当前递归级的任务节点的状态数据；

进一步的，所述基于各任务节点的状态数据及依赖关系，渲染得到节点关系树并展示的步骤包括：

基于所述数据文件中，以预置数据结构存储的当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系，渲染得到全局关系树并展示。

3、根据权利要求 1 所述的方法，其特征在于，所述基于各任务节点的状态数据及依赖关系，渲染得到全局关系树并展示的步骤，包括

以当前任务节点为根节点，根据当前任务节点和下游依赖任务节点的状态数据及依赖关系，进行正向渲染获得第一多叉树；

以指定的任务节点为根节点，根据当前任务节点和上游依赖任务节点的状态数据及依赖关系，进行反向渲染获得第二多叉树；

将第一多叉树和第二多叉树拼接为全局关系树并展示。

4、根据权利要求 3 所述的方法，其特征在于，所述以指定的任务节点为根节点，根据当前任务节点和上游依赖任务节点的状态数据及依赖关系，进行反向渲染获得第二多叉树的步骤包括：

以指定的任务节点为根节点，根据当前任务节点和下游依赖任务节点的状态数据及依赖关系，进行正向渲染获得与第一多叉树同向的第二多叉树；

将与第一多叉树同向的第二多叉树进行倒置，获得最终的第二多叉树。

5、根据权利要求 1 所述的方法，其特征在于，所述根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖任务节点和下游依赖任务节点的步骤，包括：

以当前任务节点为起点，分别采用深度优先搜索算法向上游、下游进行递归搜索，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系。

6、根据权利要求 1 所述的方法，其特征在于，所述根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖任务节点和下游依赖任务节点的步骤包括：

根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，在分布式缓存中分别递归搜索上游依赖任务节点和下游依赖任务节点；

当在分布式缓存中未能获取到上游依赖任务节点和下游依赖任务节点，则进入数据库中分别递归搜索上游依赖任务节点和下游依赖任务节点。

7、根据权利要求 1 所述的方法，其特征在于，在所述任务展示请求中还包括：

以所述当前任务节点为起点的上游依赖层级和下游依赖层级；

进一步的，所述根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖任务节点和下游依赖任务节点的步骤，包括：

根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖层级之内的上游依赖任务节点和下游依赖层级之内的下游依赖任务节点。

8、一种全局任务节点依赖关系可视化装置，其特征在于，包括：

展示请求发起模块，用于发起任务展示请求；所述任务展示请求包括当前任务节点的标识；

递归搜索模块，用于根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖任务节点和下游依赖任务节点；其中，在递归过程中，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系；

渲染展示模块，用于基于各任务节点的状态数据及依赖关系，渲染得到全局关系树

并展示。

9、根据权利要求 8 所述的装置，其特征在于，在递归搜索模块之后，还包括：

节点数据存储模块，用于采用预置数据结构，将所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系存储为数据文件；所述预置数据结构包括：第一字段：用于存储递归形式的上游依赖关系；第二字段：用于存储递归形式的下游依赖关系；第三字段：用于存储当前递归级的任务节点标识；第四字段：用于存储当前递归级的任务节点的状态数据；

进一步的，所述渲染展示模块包括：

第一渲染展示模块，用于基于所述数据文件中，以预置数据结构存储的当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系，渲染得到全局关系树并展示。

10、根据权利要求 9 所述的装置，其特征在于，所述渲染展示模块包括

第一多叉树渲染模块，用于以当前任务节点为根节点，根据当前任务节点和下游依赖任务节点的状态数据及依赖关系，进行正向渲染获得第一多叉树；

第二多叉树渲染模块，用于以指定的任务节点为根节点，根据当前任务节点和上游依赖任务节点的状态数据及依赖关系，进行反向渲染获得第二多叉树；

拼接展示模块，用于将第一多叉树和第二多叉树拼接为全局关系树并展示。

11、根据权利要求 10 所述的装置，其特征在于，所述第二多叉树渲染模块包括：

同向渲染模块，用于以指定的任务节点为根节点，根据当前任务节点和下游依赖任务节点的状态数据及依赖关系，进行正向渲染获得与第一多叉树同向的第二多叉树；

倒置模块，用于将与第一多叉树同向的第二多叉树进行倒置，获得最终的第二多叉树。

12、根据权利要求 8 所述的装置，其特征在于，所述递归搜索模块包括：

深度递归搜索模块，用于以当前任务节点为起点，分别采用深度优先搜索算法向上游、下游进行递归搜索，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系。

13、根据权利要求 8 所述的装置，其特征在于，所述递归搜索模块包括：

分布式搜索模块，用于根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，在分布式缓存中分别递归搜索上游依赖任务节点和下游依赖任务节点；

数据库搜索模块，用于当在分布式缓存中未能获取到上游依赖任务节点和下游依赖

任务节点，则进入数据库中分别递归搜索上游依赖任务节点和下游依赖任务节点。

14、根据权利要求 8 所述的装置，其特征在于，在所述任务展示请求中还包括：

以所述当前任务节点为起点的上游依赖层级和下游依赖层级；

进一步的，所述递归搜索模块包括：

层级递归搜索模块，用于根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖层级之内的上游依赖任务节点和下游依赖层级之内的下游依赖任务节点。

15、一种全局任务节点依赖关系可视化系统，其特征在于，包括：客户端和后台服务器；

所述客户端包括：

展示请求发起模块，用于发起任务展示请求；所述任务展示请求包括当前任务节点的标识；

渲染展示模块，用于基于各任务节点的状态数据及依赖关系，渲染得到全局关系树并展示；

所述后台服务器包括：

递归搜索模块，用于根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖任务节点和下游依赖任务节点；其中，在递归过程中，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系，并将各任务节点的状态数据及依赖关系返回客户端。

16、根据权利要求 15 所述的系统，其特征在于，所述后台服务器还包括：

节点数据存储模块，用于采用预置数据结构，将所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系存储为数据文件，并将所述数据文件返回客户端；所述预置数据结构包括：第一字段：用于存储递归形式的上游依赖关系；第二字段：用于存储递归形式的下游依赖关系；第三字段：用于存储当前递归级的任务节点标识；第四字段：用于存储当前递归级的任务节点的状态数据；

进一步的，所述渲染模块包括：

第一渲染模块，用于基于所述数据文件中，以预置数据结构存储的当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系，渲染得到全局关系树并展示。

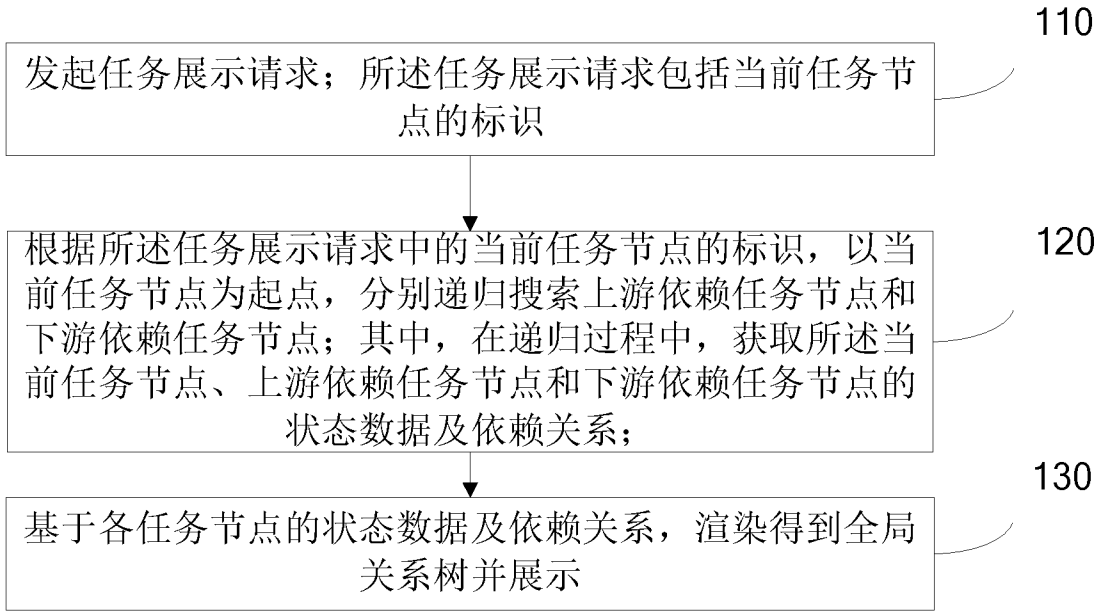
发起任务展示请求；所述任务展示请求包括当前任务节点的标识
110
根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖任务节点和下游依赖任务节点；其中，在递归过程中，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系；
120
基于各任务节点的状态数据及依赖关系，渲染得到全局关系树并展示
130

图 1

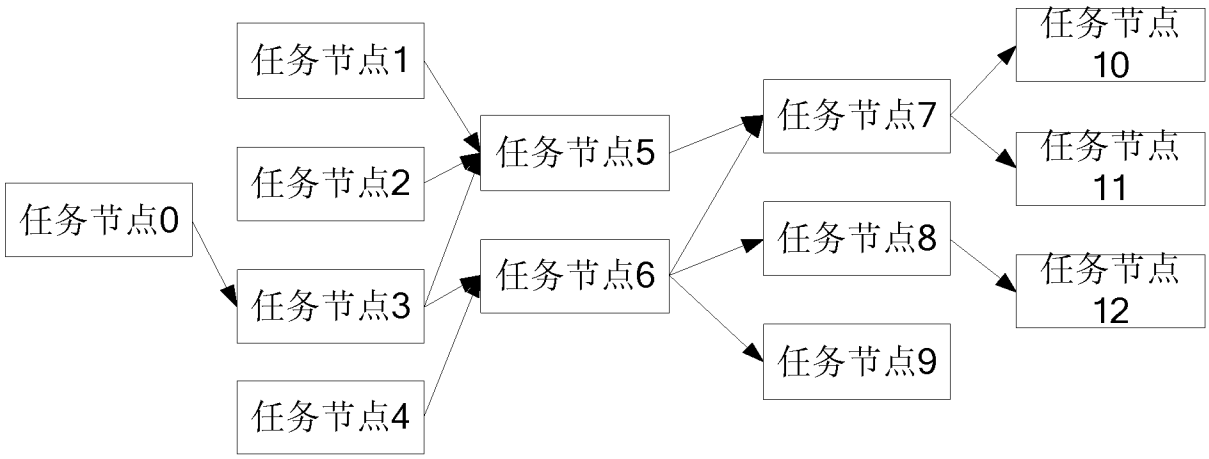
任务节点0
任务节点1
任务节点2
任务节点3
任务节点4
任务节点5
任务节点6
任务节点7
任务节点8
任务节点9
任务节点10
任务节点11
任务节点12

图 1A

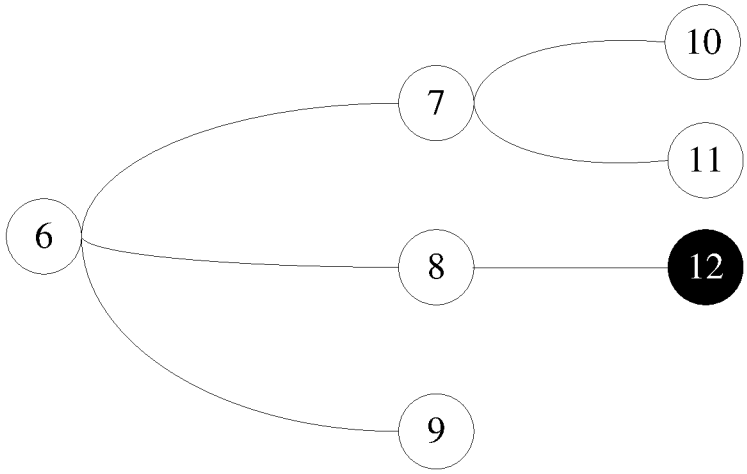

图 1B

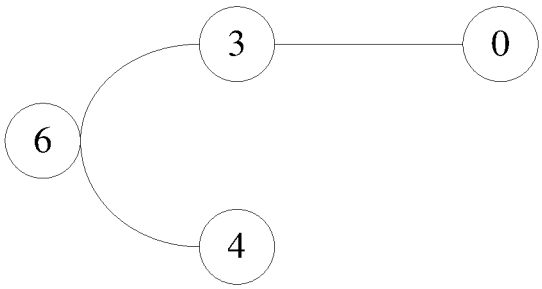

图 1C

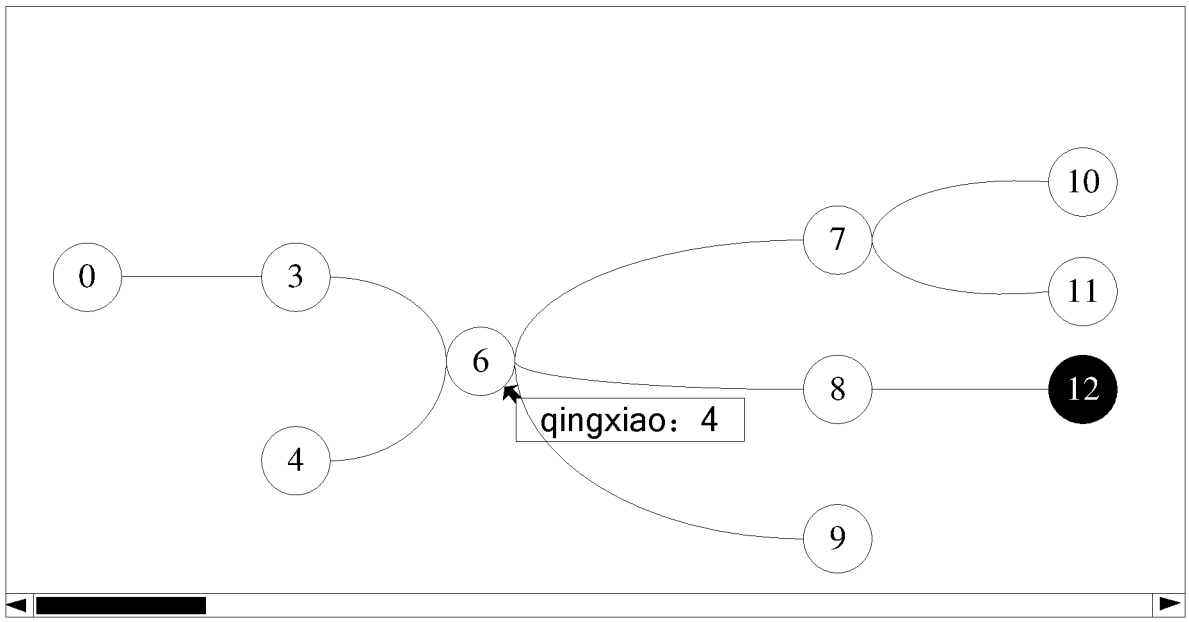

图 1D

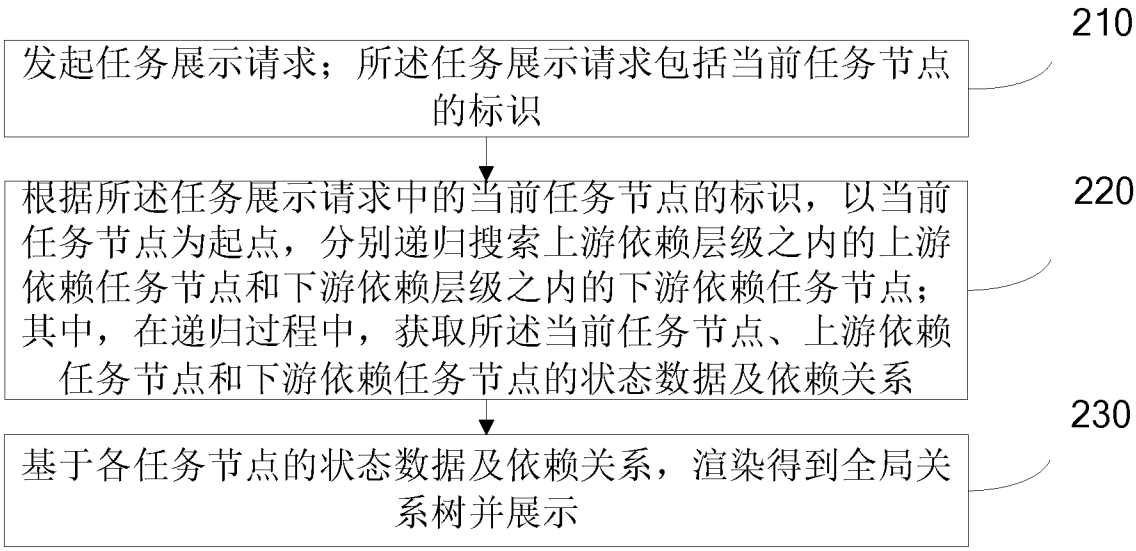
210
发起任务展示请求；所述任务展示请求包括当前任务节点的标识
220
根据所述任务展示请求中的当前任务节点的标识，以当前任务节点为起点，分别递归搜索上游依赖层级之内的上游依赖任务节点和下游依赖层级之内的下游依赖任务节点；其中，在递归过程中，获取所述当前任务节点、上游依赖任务节点和下游依赖任务节点的状态数据及依赖关系
230
基于各任务节点的状态数据及依赖关系，渲染得到全局关系树并展示

图 2


展示请求发起模块310
递归搜索模块320
渲染展示模块330

图 3


展示请求发起模块410
递归搜索模块420
层级递归搜索模块422
渲染展示模块430

图 4

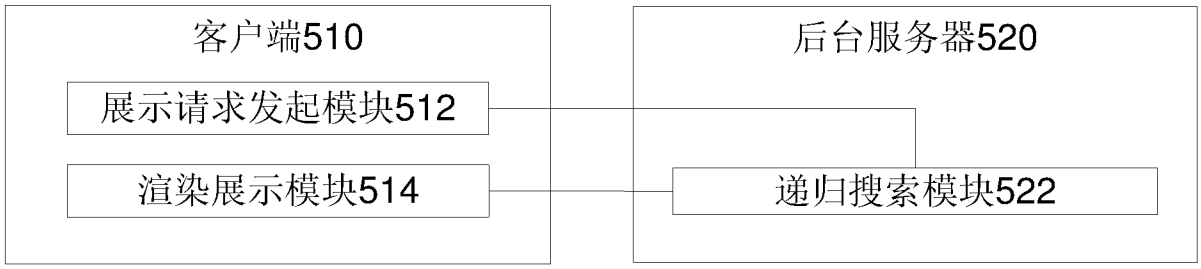

图 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2016/083889

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/50 (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)

G06F; H04L

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)

WPI, EPODOC, CNPAT, CNKI, IEEE: ODPS, overall situation, recursion, upstream, downstream, mission, task, assigment, node, rely, dependent, logic, father, children, recursion, tree, status, display, visual+, show

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CN 103019691 A (BEIJING SI-TECH INFORMATION TECHNOLOGY CO., LTD.) 03 April 2013 (03.04.2013) description, paragraphs [0002], [0018], and [0028]-[0045]	1-16
A	CN 101702157 A (KINGDEE SOFTWARE (CHINA) CO., LTD.) 05 May 2010 (05.05.2010) the whole document	1-16
A	CN 1713192 A (IBM) 28 December 2005 (28.12.2005) the whole document	1-16
A	US 2003131222 A1 (THOMAS III, PAUL AUGUSTUS) 10 July 2003 (10.07.2003) the whole 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

"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
04 August 2016	26 August 2016
Name and mailing address of the ISA State Intellectual Property Office of the P. R. China No. 6, Xitucheng Road, Jimenqiao Haidian District, Beijing 100088, China Facsimile No. (86-10) 62019451	Authorized officer JU, Bo Telephone No. (86-10) 62413661

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2016/083889

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
CN 103019691 A	03 April 2013	None	
CN 101702157 A	05 May 2010	None	
CN 1713192 A	28 December 2005	US 2008262815 A1	23 October 2008
		US 2005289088 A1	29 December 2005
US 2003131222 A1	10 July 2003	EP 1327933 A2	16 July 2003

国际检索报告

国际申请号
PCT/CN2016/083889

A. 主题的分类

G06F 9/50(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)

G06F H04L

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））

WPI,EPODOC,CNPAT,CNKI,IEEE:ODPS,任务,节点,依赖,依靠,逻辑,全局,递归,上游,下游, 父亲,孩子,树,状态,可视,显示,展示,mission,task,assigment,node,rely,dependent, logic,father,children,recursion,tree,status,display,visual+

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
X	CN 103019691 A (北京思特奇信息技术股份有限公司) 2013年 4月 3日 (2013-04-03) 说明书第0002、0018、0028-0045段	1-16
A	CN 101702157 A (金蝶软件中国有限公司) 2010年 5月 5日 (2010-05-05) 全文	1-16
A	CN 1713192 A (国际商业机器公司) 2005年 12月 28日 (2005-12-28) 全文	1-16
A	US 2003131222 A1 (THOMAS III, PAUL AUGUSTUS) 2003年 7月 10日 (2003-07-10) 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

"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
2016年 8月 4日	2016年 8月 26日

ISA/CN的名称和邮寄地址	受权官员
中华人民共和国国家知识产权局(ISA/CN) 中国北京市海淀区蓟门桥西土城路6号 100088	鞠博
传真号 (86-10)62019451	电话号码 (86-10)62413661

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2016/083889

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)
CN	103019691	A	2013年 4月 3日	无			
CN	101702157	A	2010年 5月 5日	无			
CN	1713192	A	2005年 12月 28日	US	2008262815	A1	2008年 10月 23日
				US	2005289088	A1	2005年 12月 29日
US	2003131222	A1	2003年 7月 10日	EP	1327933	A2	2003年 7月 16日