(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 115712524 A
(43)申请公布日 2023.02.24

(21)申请号 202211402105.0

(22)申请日 2022.11.09

(71)申请人 上海哔哩哔哩科技有限公司
地址 200433 上海市杨浦区政立路485号国正中心3号楼

(72)发明人 马阳阳 丁国涛 周致达

(74)专利代理机构 北京英特普罗知识产权代理有限公司 11015
专利代理师 周建达

(51)Int.Cl.
*G06F 11/14*(2006.01)
*G06F 9/455*(2006.01)
*G06F 9/50*(2006.01)
*G06F 9/54*(2006.01)

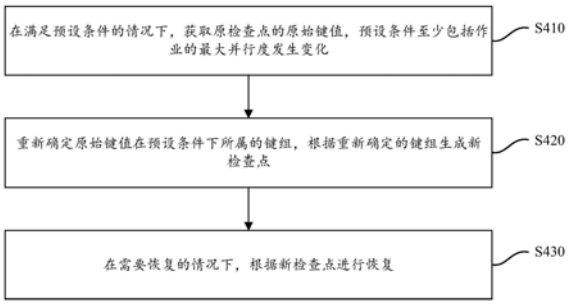
权利要求书2页 说明书11页 附图6页

(54)发明名称

# 数据恢复方法及装置

(57)摘要

本申请实施例提供一种数据恢复方法，所述方法包括：在满足预设条件的情况下，获取原检查点的原始键值，所述预设条件至少包括作业的最大并行度发生变化；重新确定所述原始键值在所述预设条件下所属的键组，并根据重新确定的所述键组生成新检查点；在需要恢复的情况下，根据所述新检查点进行恢复。本申请实施例提供的数据恢复方法，可以在作业的最大并行度发生变化的情况下进行检查点的恢复，提高了检查点的可恢复性。




CN 115712524 A

1.一种数据恢复方法,其特征在于,包括:

在满足预设条件的情况下,获取原检查点的原始键值,所述预设条件至少包括作业的最大并行度发生变化;

重新确定所述原始键值在所述预设条件下所属的键组,并根据重新确定的所述键组生成新检查点;

在需要恢复的情况下,根据所述新检查点进行恢复。

2.根据权利要求1所述的数据恢复方法,其特征在于,所述获取原检查点的原始键值,包括:

从原状态文件中获取原检查点的第一键值,所述第一键值为经过序列化的所述原始键值;

对所述第一键值中的键进行反序列化,将反序列化的结果作为所述原始键值。

3.根据权利要求2所述的数据恢复方法,其特征在于,所述从原状态文件中获取原检查点的第一键值,包括:

通过状态处理器API从原状态文件中获取原检查点的第一键值。

4.根据权利要求2所述的数据恢复方法,其特征在于,所述从原状态文件中获取原检查点的第一键值,包括:

从元数据文件中读取原检查点所有算子的第一ID;

对每个所述第一ID依次调用状态处理器API以获取所述第一键值。

5.根据权利要求4所述的数据恢复方法,其特征在于,在所述对每个所述第一ID依次调用状态处理器API以获取所述第一键值之前,还包括:

在所述元数据文件中加入第一信息;

所述对每个所述第一ID依次调用状态处理器API以获取所述第一键值,包括:

从所述元数据文件中获取所述第一信息;

根据所述第一信息生成状态描述符;

根据所述状态描述符对每个所述第一ID调用状态处理器API以获取所述第一键值。

6.根据权利要求1-5任一项所述的数据恢复方法,其特征在于,还包括:

调整算子的最大并行度的计算方式,以增大所述算子的最大并行度。

7.根据权利要求1-5任一项所述的数据恢复方法,其特征在于,还包括:

在满足所述预设条件的情况下,根据预定规则生成算子的ID,所述预定规则包括不将上游算子与下游算子chain在一起。

8.根据权利要求7所述的数据恢复方法,其特征在于,还包括:

在所述算子的ID发生变化的情况下,获取所述算子名称和当前作业的有向无环图中算子的第二ID;

根据所述算子名称关联所述第一ID和所述第二ID;

根据所述第一ID与所述第二ID的关联关系,将所述原检查点的状态数据分配至所述有向无环图的算子中。

9.一种数据恢复装置,其特征在于,包括:

获取模块,用于在满足预设条件的情况下,获取原检查点的原始键值,所述预设条件至少包括作业的最大并行度发生变化;

确定模块，用于重新确定所述原始键值在所述预设条件下所属的键组，并根据重新确定的所述键组生成新检查点；

恢复模块，用于在需要恢复的情况下，根据所述新检查点进行恢复。

10.一种计算机设备，所述计算机设备包括存储器、处理器以及存储在存储器上并可在处理器上运行的计算机程序，其特征在于，所述处理器执行所述计算机程序时用于实现权利要求1至8中任一项所述的数据恢复方法的步骤。

11.一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内存储有计算机程序，所述计算机程序可被至少一个处理器所执行，以使所述至少一个处理器执行权利要求1至8中任一项所述的数据恢复方法的步骤。

# 数据恢复方法及装置

## 技术领域

[0001] 本申请涉及大数据技术领域,特别涉及一种数据恢复方法、装置、计算机设备及存储介质。

## 背景技术

[0002] Flink是由Apache软件基金会开发的开源流处理框架,提供高吞吐量、低延迟的流数据引擎以及对事件-时间处理和状态管理的支持。为了实现容错,Flink将状态数据(state)进行存储,从而方便在发生异常时可以进行恢复。

[0003] 目前,以Flink为基础的实时计算广泛而深入地应用在越来越多的平台中。在Flink作业中,随着流量或者计算复杂度变化,用户或者平台可能需要改变作业的最大并行度以增加处理能力。然而,最大并行度的变化后,Flink无法正常进行检查点(checkpoint)的恢复。

## 发明内容

[0004] 本申请的目的在于提供一种数据恢复方法、装置、计算机设备及存储介质,用于解决目前因最大并行度变化导致无法正常进行检查点恢复的问题。

[0005] 本申请实施例的一个方面提供了一种数据恢复方法,包括:在满足预设条件的情况下,获取原检查点的原始键值,所述预设条件至少包括作业的最大并行度发生变化;重新确定所述原始键值在所述预设条件下所属的键组,并根据重新确定的所述键组生成新检查点;在需要恢复的情况下,根据所述新检查点进行恢复。

[0006] 可选地,所述获取原检查点的原始键值,包括:从原状态文件中获取原检查点的第一键值,所述第一键值为经过序列化的所述原始键值;对所述第一键值中的键进行反序列化,将反序列化的结果作为所述原始键值。

[0007] 可选地,所述从原状态文件中获取原检查点的第一键值,包括:通过状态处理器API从原状态文件中获取原检查点的第一键值。

[0008] 可选地,所述从原状态文件中获取原检查点的第一键值,包括:从元数据文件中读取原检查点所有算子的第一ID;对每个所述第一ID依次调用状态处理器API以获取所述第一键值。

[0009] 可选地,在所述对每个所述第一ID依次调用状态处理器API以获取所述第一键值之前,还包括:在所述元数据文件中加入第一信息;所述对每个所述第一ID依次调用状态处理器API以获取所述第一键值,包括:从所述元数据文件中获取所述第一信息;根据所述第一信息生成状态描述符;根据所述状态描述符对每个所述第一ID调用状态处理器API以获取所述第一键值。

[0010] 可选地,方法还包括:调整算子的最大并行度的计算方式,以增大所述算子的最大并行度。

[0011] 可选地,方法还包括:在满足所述预设条件的情况下,根据预定规则生成算子的

ID,所述预定规则包括不将上游算子与下游算子chain在一起。

[0012] 可选地,方法还包括:在所述算子的ID发生变化的情况下,获取所述算子名称和当前作业的有向无环图中算子的第二ID;根据所述算子名称关联所述第一ID和所述第二ID;根据所述第一ID与所述第二ID的关联关系,将所述原检查点的状态数据分配至所述有向无环图的算子中。

[0013] 本申请实施例的一个方面又提供了一种数据恢复装置,包括:获取模块,用于在满足预设条件的情况下,获取原检查点的原始键值,所述预设条件至少包括作业的最大并行度发生变化;确定模块,用于重新确定所述原始键值在所述预设条件下所属的键组,并根据重新确定的所述键组生成新检查点;恢复模块,用于在需要恢复的情况下,根据所述新检查点进行恢复。

[0014] 本申请实施例的一个方面又提供了一种计算机设备,所述计算机设备包括存储器、处理器以及存储在存储器上并可在处理器上运行的计算机程序,所述处理器执行所述计算机程序时用于实现上述的数据恢复方法的步骤。

[0015] 本申请实施例的一个方面又提供了一种计算机可读存储介质,所述计算机可读存储介质内存储有计算机程序,所述计算机程序可被至少一个处理器所执行,以使所述至少一个处理器执行上述的数据恢复方法的步骤。

[0016] 本申请实施例提供的数据恢复方法、装置、计算机设备及存储介质,包括以下优点:

[0017] 通过在满足预设条件(至少包括作业的最大并行度发生变化)的情况下,获取原检查点的原始键值,重新确定原始键值在预设条件下所属的键组,并根据重新确定的键组生成新检查点,在需要恢复的情况下,根据新检查点进行相应的恢复操作;由于在最大并行度发生变化的情况下原始键值所属的键组会发生变化,因此获取原始键值并重新确定原始键值所属的键组,即可以根据重新确定的键组生成新检查点,方便根据新状态文件中新检查点对应的状态数据进行恢复,从而实现在最大并行度发生变化的情况下进行检查点恢复,提高了检查点的可恢复性。

## 附图说明

[0018] 图1示意性示出了本申请实施例的环境架构图;

[0019] 图2示意性示出了本申请实施例的数据恢复方法的应用环境图;

[0020] 图3示意性示出了本申请实施例一的数据恢复方法的流程图;

[0021] 图4为图3中步骤S410的子步骤的流程图;

[0022] 图5为图4中步骤S411的子步骤的流程图;

[0023] 图6为图5中步骤S520的子步骤的流程图;

[0024] 图7为数据恢复方法的原理示例图;

[0025] 图8为图3新增步骤的流程图;

[0026] 图9示意性示出了本申请实施例二的数据恢复装置的框图;

[0027] 图10示意性示出了本申请实施例三的计算机设备的硬件架构图。

## 具体实施方式

[0028] 为了使本申请的目的、技术方案及优点更加清楚明白，以下结合附图及实施例，对本申请进行进一步详细说明。应当理解，此处所描述的具体实施例仅用以解释本申请，并不用于限定本申请。基于本申请中的实施例，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，都属于本申请保护的范围。

[0029] 需要说明的是，在本申请实施例中涉及“第一”、“第二”等的描述仅用于描述目的，而不能理解为指示或暗示其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。另外，各个实施例之间的技术方案可以相互结合，但是必须是以本领域普通技术人员能够实现为基础，当技术方案的结合出现相互矛盾或无法实现时应当认为这种技术方案的结合不存在，也不在本申请要求的保护范围之内。

[0030] 在本申请的描述中，需要理解的是，步骤前的数字标号并不标识执行步骤的前后顺序，仅用于方便描述本申请及区别每一步骤，因此不能理解为对本申请的限制。

[0031] 下面为本申请涉及的术语解释：

[0032] Flink，是由Apache软件基金会开发的开源流处理框架，其核心是用Java和Scala编写的分布式流数据处理引擎，以数据并行和管道方式执行任意流数据程序，其流水线运行时系统可以执行批处理和流处理程序。

[0033] Kafka，是由Apache软件基金会开发的一个开源流处理平台，其目标是为处理实时数据提供一个统一、高吞吐、低延迟的平台，其持久化层本质上是一个“按照分布式事务日志架构的大规模发布/订阅消息队列”，其可以通过Kafka Connect连接到外部系统(用于数据输入/输出)，并提供了Kafka Streams(一个Java流式处理库)。

[0034] Kubernetes(常简称为K8s)，是用于自动部署、扩展和管理“容器化(containerized)应用程序”的开源系统。

[0035] Yarn(Yet Another Resource Negotiator，另一种资源协调者)，是一种新的Hadoop资源管理器，它是一个通用资源管理系统，可为上层应用提供统一的资源管理和调度，它的引入为集群在利用率、资源统一管理和数据共享等方面带来了巨大好处。

[0036] HDFS，是Hadoop的分布式文件系统(Hadoop Distributed File System)，实现大规模数据可靠的分布式读写。

[0037] Zookeeper，是Apache软件基金会的一个软件项目，它为大型分布式计算提供开源的分布式配置服务、同步服务和命名注册。

[0038] RocksDB，是用于键值数据的高性能嵌入式数据库，是谷歌LevelDB的一个分支，经过优化，可以利用多核CPU，并有效利用固态驱动器(SSD)等快速存储来处理输入/输出(I/O)绑定的工作负载，使用了基于日志结构的合并树(LSM树)数据结构。

[0039] 图1示意性示出了本申请实施例的环境架构图，如图所示：

[0040] 客户端100与计算引擎200相连接，计算引擎200与状态后端300相连接。用户可以通过客户端100改变计算引擎200作业的最大并行度；计算引擎200将状态数据存储至状态后端300进行检查点的操作，在发生故障时，根据检查点进行恢复，实现容错。本申请实施例的数据恢复方法，在满足预设条件(包括计算引擎200的作业的最大并行度发生变化)，从状态后端300的原状态文件中获取原检查点的原始键值，重新确定原始键值在预设条件下所

属的键组(key-group),并根据重新确定的键组在状态后端300生成新检查点;在需要恢复的情况下,计算引擎200根据生成的新检查点进行恢复。其中,计算引擎200可以为Flink,状态后端300可以为但不限于RocksDB。

[0041] 请参考图2,其为本申请实施例的数据恢复方法的应用环境示例图。如图所示:

[0042] Flink作为计算引擎200可以运行在若干种集群环境中,例如,纯物理机部署的Yarn集群、与Kafka混合部署的Yarn集群(为了提高Kafka的资源利用率)、或者与K8S混合部署的Yarn集群(为了提高线上服务器的资源利用率)等。

[0043] 本申请实施例的数据恢复方法中,Flink在满足预设条件(包括作业的最大并行度发生变化)时,获取原检查点的原始键值,重新确定原始键值在预设条件下所属的键组,根据重新确定的键组生成新检查点;在需要恢复的情况下,根据生成的新检查点进行恢复。

[0044] 在相关技术中,用户或者平台改变作业的最大并行度来增加处理能力时,但作业的最大并行度变化后,Flink无法正常进行检查点的恢复。

[0045] 本申请实施例的数据恢复方案,可以在作业的最大并行度发生变化的情况下进行检查点的恢复。

[0046] 以下将通过若干个实施例介绍数据恢复方案,为便于理解,下面将以图1中的计算引擎200为执行主体进行示例性描述。在某些实施例中,计算引擎200直接以Flink作为示例进行描述。

[0047] 实施例一

[0048] 图3示意性示出了本申请实施例的数据恢复方法的流程图,如图所示,可以包括步骤S410～步骤S430,具体说明如下:

[0049] 步骤S410,在满足预设条件的情况下,获取原检查点的原始键值,预设条件至少包括作业的最大并行度发生变化。

[0050] 作业的最大并行度发生变化包括最大并行度增大或缩小。在作业的最大并行度发生变化的情况下,Flink本身的执行逻辑导致无法进行状态的分配,从而无法正常进行检查点的恢复。

[0051] 可以理解的是,步骤S410～步骤S430是在满足预设条件的情况下执行的步骤,而预设条件除了最大并行度发生变化之外,还可以根据实际需要设置其它的条件,此处不做具体限制。

[0052] 为了从状态恢复时,不必拉取所有的状态文件,Flink使用了类似一致性哈希的做法,将状态的键值做哈希后划分到一个固定的份数里,每个算子的一个并行度负责其中的一个范围。Flink在获取原检查点的原始键值时,可以根据最大并行度改变之前算子对应的ID从原状态文件中获取原检查点的原始键值。

[0053] 步骤S420,重新确定原始键值在预设条件下所属的键组,根据重新确定的键组生成新检查点。

[0054] 在作业的最大并行度发生变化的情况下,算子的数量会发生变化,原始键值需要重新分配至变化后的算子,从而使得原始键值所属的键组(key-group)发生变化,需要重新确定每个算子所属的键组。例如,在作业的最大并发度增大的情况下,算子的数量会增多,原始键值需要重新分配至更多数量的算子中,至少部分算子负责的键会变少,此时需要确定每个算子在变化后对应的键,从而重新确定原始键值所属的键组。

[0055] 在重新确定原始键值在预设条件下所属的键组时，可以根据变化后的最大并行度将原始键值重新分配至在预设条件下的算子，从而确定原始键值在预设条件下所属的键组。其中，在将原始键值重新分配至算子时，是指将原始键值中的键(key)重新分配至算子。例如，假设最大并行度变化前有2个算子，原始键值中的键为1-10，最大并行度变化后有5个算子，则是将键1-2分配给0号算子，3-4分配至1号算子，5-6分配至2号算子，7-8分配至3号算子，9-10分配至4号算子，根据分配后每个算子对应的键确定在最大并行度增大后每个键所属的键组。

[0056] 在重新确定原始键值所属的键组之后，即可以将新的键组写入新的状态文件中，从而生成新检查点。

[0057] 步骤S430，在需要恢复的情况下，根据新检查点进行恢复。

[0058] 由于已根据重新确定的键组生成新检查点，因此当计算引擎200发生故障需要恢复时，可以从新的状态文件中获取新检查点对应的状态数据来进行恢复。

[0059] 具体地，可以是在接收到恢复的指令时，Flink根据恢复的指令和新检查点进行恢复，从而将状态恢复至新检查点，实现容错。

[0060] 本申请实施例提供的数据恢复方法，通过在满足预设条件(至少包括作业的最大并行度发生变化)的情况下，获取原检查点的原始键值，重新确定原始键值在预设条件下所属的键组，并根据重新确定的键组生成新检查点，在需要恢复的情况下，根据新检查点进行相应的恢复操作；由于在最大并行度发生变化的情况下原始键值所属的键组会发生变化，因此获取原始键值并重新确定原始键值所属的键组，即可以根据重新确定的键组生成新检查点，方便根据新状态文件中新检查点对应的状态数据进行恢复，从而实现在最大并行度发生变化的情况下进行检查点恢复，提高了检查点的可恢复性。

[0061] 在示例性的实施例中，如图4所示，步骤S410中，获取原检查点的原始键值，可以包括步骤S411～步骤S412，具体如下：

[0062] 步骤S411，从原状态文件中获取原检查点的第一键值，第一键值为经过序列化的原始键值。

[0063] 由于原始键值在写入原状态文件中时，会经过序列化，因此在原状态文件中获取原检查点的键值时，获取到的是经过序列化的原始键值，亦即第一键值。

[0064] 步骤S412，对第一键值中的键进行反序列化，将反序列化的结果作为原始键值。

[0065] 可选地，计算引擎200可以对第一键值中的键和值同时进行反序列化，再根据反序列化的键和值得到原始键值。然而，由于原始键值中的值包括的数据量通常较大，因此如果对原始键值的值进行反序列化，将会占用较多的计算资源，影响计算引擎200的性能；并且，重新确定原始键值在预设条件下所属的键组时，只需要根据键就可以确定所属的键组，因此，可以只对第一键值中的键进行反序列化，而不对第一键值中的值进行反序列化。

[0066] 具体地，可以将第一键值中的值设置为不需要反序列化的对象，如可以将value作为字节数组看待，从而只反序列化第一键值中的键而不反序列化第一键值中的值。

[0067] 在本实施例中，通过从原状态文件中获取原检查点的第一键值，再对第一键值中的键进行反序列化，将反序列化的结果作为原始键值，可以只反序列化第一键值中的键而不反序列化第一键值中的值，避免反序列化第一键值中的值过多占用计算资源而影响性能，提高了获取原始键值的效率。

[0068] 计算引擎200在从原状态文件中获取原检查点的第一键值时，可以通过预设工具来获取。在示例性的实施例中，步骤S411中，从原状态文件中获取原检查点的第一键值，可以包括：通过状态处理器API(State Processor API，状态处理器应用程序接口)从原状态文件中获取原检查点的第一键值。

[0069] 亦即，预设工具为状态处理器API。其中，状态处理器API可以读取、写入或修改保存点(savepoint)和检查点，也可以转为SQL(结构化查询语言)查询来分析和处理状态数据。由于状态处理器API是社区原生的，因此通过状态处理器API从原状态文件中获取原检查点的第一键值，可以降低获取原始键值的实现难度。

[0070] 由于状态处理器API提供的用户接口是基于单个算子的，而实际上需要对原检查点所有的状态进行读取和计算，因此需要对状态处理器API进行适当的改造。在示例性的实施例中，步骤S411中，从原状态文件中获取原检查点的第一键值，如图5所示，可以包括步骤S510～步骤S520，具体如下：

[0071] 步骤S510，从元数据文件中读取原检查点所有算子的第一ID。

[0072] 步骤S520，对每个第一ID依次调用状态处理器API以获取第一键值。

[0073] 具体地，由于元数据文件中包括所有算子的ID，因此Flink可以从元数据文件中读取原检查点所有算子的ID(即第一ID)，再根据每个第一ID依次调用状态处理器API对原检查点每个算子进行计算，可以得到原检查点所有算子对应的第一键值。

[0074] 本实施例中，从元数据文件中读取原检查点所有算子的第一ID，对每个第一ID依次调用状态处理器API以获取第一键值，由于原检查点中每个算子对应的第一键值均可以通过元数据文件中的第一ID调用状态处理器API来获取，因此可以利用状态处理器API获取到原检查点所有算子的第一键值，克服状态处理器API提供的用户接口只能基于单个算子的缺陷。

[0075] 在前述实施例中，步骤S412中，对第一键值中的键进行反序列化，由于状态处理器API也可以对第一键值进行反序列化，因此也可以通过状态处理器API来实现对第一键值中的键进行反序列化。另外，由于状态处理器API会同时反序列化第一键值中的键和值，因此可以通过对第一键值中的值进行特殊处理来只反序列化第一键值中的键而不反序列化第一键值中的值，例如上述的将第一键值中的值作为字节数组来看待；也可以设置状态处理器API直接跳过第一键值中的值的反序列化过程，从而只对第一键值中的键进行反序列化。

[0076] 在示例性的实施例中，在步骤S520之前，即在对每个第一ID依次调用状态处理器API以获取第一键值之前，还可以包括：在元数据文件中加入第一信息；相应地，如图6所示，步骤S520，可以包括步骤S521～步骤S523，具体如下：

[0077] 步骤S521，从元数据文件中获取第一信息。

[0078] 第一信息可以包括但不限于状态数据名称、序列化器、状态数据类型等信息。其中，状态数据可以包括：

[0079] Value状态数据：即类型为T的单值状态，与对应的key绑定，可以通过update方法更新状态值，通过value()方法获取状态值；

[0080] List状态数据：即key上的状态值为一个列表，可以通过add方法往列表中附加值，也可以通过get()方法返回一个Iterable<T>来遍历状态值；

[0081] Reducing状态数据：这种状态通过用户传入的reduceFunction，每次调用add方法

添加值的时候，会调用reduceFunction，最后合并到一个单一的状态值；

[0082] Folding状态数据：跟Reducing状态数据类似；

[0083] Map状态数据：即状态值为一个map，通过put或putall方法添加元素。

[0084] 步骤S522，根据第一信息生成状态描述符(StateDescriptor)。

[0085] 由于Flink通过状态描述符来定义一个状态，而状态描述符为一个抽象类，其内部定义了状态数据名称、序列化器、状态数据类型等基础信息，因此根据第一信息生成状态描述符，可以生成调用状态处理器API用户接口需要的状态描述符。与上面的状态数据相对应，状态描述符派生了Value状态描述符、List状态描述符等描述符。

[0086] 步骤S523，根据状态描述符对每个第一ID调用状态处理器API以获取第一键值。

[0087] 在生成调用状态处理器API用户接口的状态描述符，Flink即可以根据状态描述符对每个第一ID调用状态处理器API，从而获取到相应的第一键值。

[0088] 由于目前的状态处理器API需要用户手动构造状态描述符并传入相应的方法中，才能生成调用状态处理器API用户接口需要的状态描述符，因此效率较低。而本实施例中，从元数据文件中获取预先加入的第一信息，根据第一信息生成状态描述符，根据状态描述符对每个第一ID调用状态处理器API以获取第一键值，由于可以自动根据预先加入的第一信息生成状态描述符，因此可以有效地提高利用状态处理器API获取第一键值的效率。

[0089] 请参考图7，其为数据恢复方法的原理示例图。具体地，可以预先在元数据文件(metadata)中加入第一信息(对应图中的WriteStateDescriptor)，另外，checkpointCoordinator(检查点协调器)会在元数据文件(_metadata)中写入算子的ID和状态句柄(对应图中的WriteOperatorIDsandStateHandles)，JobManager(作业管理器)从元数据文件(metadata)中读取第一信息，根据第一信息生成相应的状态描述符；并从元数据文件(_metadata)获取算子的ID，根据状态描述符对每个算子的ID调用状态处理器API获取原始键值，在重新确定原始键值所属的键组后，将重新确定的键组写入至元数据文件，并根据重新确定的键组生成新检查点，从而可以根据新检查点进行恢复，提高检查点的可恢复性。

[0090] 在示例性的实施例中，数据恢复方法还可以包括：调整算子的最大并行度的计算方式，以增大算子的最大并行度。

[0091] 可选地，由于Flink对算子的最大并行度的算法中，会设置算子的最大并行度的最小值，因此调整算子最大并行度的计算方式可以包括增大算子的最大并行度的最小值。例如，若算子原来最大并行度的最小值为128，则可以调整为1024。另外，还可以在算子最大并行度的算法公式中乘以N(N大于1)，从而将算子最大并行度增大为原来的N倍，例如，可以将原来算子最大并行度的算法公式中整体乘以10，从而将算子最大并行度增大为原来的10倍。

[0092] 可以理解的是，当算子的最大并行度较小时，则可能会经常由于流量变化或者复杂度变化而需要改变最大并行度，从而需要重新生成新的检查点来恢复。本实施例中，通过调整算子最大并行度的计算方式，以增大最大并行度，可以在一开始就使用比较合适的并行度对作业进行处理，减少因流量变化或者复杂度变化而需要改变最大并行度，进而需要生成新检查点，根据新检查点进行恢复的情况。另外，在实际应用中，可以对新的作业采用增大并行度的方法，而对于存量作业可以采用前述实施例对应的方法进行检查点的恢复。

[0093] 在示例性的实施例中，数据恢复方法还可以包括：在满足预设条件的情况下，根据预定规则生成算子的ID，其中预定规则包括不将上游算子与下游算子chain在一起。

[0094] 目前，Flink通常运行在以下集群环境中：纯物理机部署的Yarn集群、与Kafka混合部署的Yarn集群、与K8S混合部署的Yarn集群。若作业的最大并行度发生变化，则可能导致KafkaSource算子和下游算子的连接关系发生变化。而在Flink社区原生的算子ID生成算法中，计算一个算子的ID时，会根据其下游可以chain(串)在一起的算子来计算；当作业的最大并行度发生变化时，由于Kafka Source算子可能从可以与下游算子chain在一起变为不能chain，因此会导致根据Flink社区原生的算子ID生成算法计算出的算子ID发生变化，从而无法从原检查点中恢复。

[0095] 可选地，也可以直接根据预定规则生成算子的ID，而不管是否满足预设条件。另外，在生成上游算子的ID时，不考虑与其chain在一起的算子的ID，具体算法可以根据实际需要设置，此处不做限制。例如，可以对Flink社区原生的算子ID生成算法StreamGraphHasherV2进行拓展，在拓展的算法中，通过不将上游算子和下游算子chain在一起的方式来生成算子的ID。

[0096] 在本实施例中，在满足预设条件的情况下，根据预定规则生成算子的ID，由于预定规则包括不将上游算子与下游算子chain在一起，因此生成算子的ID时，可以使算子的ID不受算子与下游算子的连接关系而发生变化，生成稳定的算子ID，方便根据算子ID进行检查点以及相应的恢复操作。

[0097] 在上面的实施例中，根据预定规则生成算子的ID，在应用到新的作业中时，不会发生无法正常进行检查点的操作的问题；但对于存量作业而言，则会使前后算子的ID发生变化，可能会存在ID冲突的情况，从而无法正常进行检查点的操作。在示例性的实施例中，如图8所示，数据恢复方法还可以包括步骤S610～步骤S630，具体如下：

[0098] 步骤S610，在算子的ID发生变化的情况下，获取算子名称和当前作业的有向无环图(DAG)中算子的第二ID。

[0099] 步骤S620，根据算子名称关联第一ID和第二ID。

[0100] 步骤S630，根据第一ID与第二ID的关联关系，将原检查点的状态数据分配至有向无环图的算子中。

[0101] 具体地，Flink在算子的ID发生变化的情况下，获取每个算子的名称和当前作业的DAG中算子的第二ID，由于算子名称基本上不存在相同的情况，因此根据算子名称来关联第一ID和第二ID，就可以将原检查点的状态数据分配至DAG的算子中，方便后续生成新检查点。实际应用中，由于部分算子不存在状态数据，而有状态数据的算子名称通常情况下不会相同，因此可以只通过名称关联有状态数据的算子，亦可以实现状态数据的分配。

[0102] 本实施例中，在算子的ID发生变化的情况下，获取算子名称和当前作业的有向无环图中算子的第二ID，根据算子名称关联第一ID和第二ID，再根据第一ID和第二ID的关联关系，将原检查点的状态分配至有向无环图的算子中，可以在算子的ID发生变化的情况下，根据算子名称将状态数据分配至DAG的算子中。

[0103] 实施例二

[0104] 图9示意性示出了根据本申请实施例二的数据恢复装置700的框图，该数据恢复装置700可以被分割成一个或多个程序模块，一个或者多个程序模块被存储于存储介质中，并

由一个或多个处理器所执行，以完成本申请实施例。本申请实施例所称的程序模块是指能够完成特定功能的一系列计算机程序指令段，以下描述将具体介绍本实施例中各程序模块的功能。

[0105] 如图9所示，该数据恢复装置700可以包括获取模块710、确定模块720和恢复模块730。

[0106] 获取模块710，用于在满足预设条件的情况下，获取原检查点的原始键值，预设条件至少包括作业的最大并行度发生变化；

[0107] 确定模块720，用于重新确定原始键值在预设条件下所属的键组，并根据重新确定的键组生成新检查点；

[0108] 恢复模块730，用于在需要恢复的情况下，根据新检查点进行恢复。

[0109] 在示例性的实施例中，获取模块710还用于：从原状态文件中获取原检查点的第一键值，第一键值为经过序列化的原始键值；对第一键值中的键进行反序列化，将反序列化的结果作为原始键值。

[0110] 在示例性的实施例中，获取模块710还用于：通过状态处理器API从原状态文件中获取原检查点的第一键值。

[0111] 在示例性的实施例中，获取模块710还用于：从元数据文件中读取原检查点所有算子的第一ID；对每个第一ID依次调用状态处理器API以获取第一键值。

[0112] 在示例性的实施例中，数据恢复装置700还包括加入模块(图中未示出)，获取模块710还用于：从元数据文件中获取第一信息；根据第一信息生成状态描述符；根据状态描述符对每个第一ID调用状态处理器API以获取第一键值。

[0113] 在示例性的实施例中，数据恢复装置700还包括调整模块(图中未示出)，其中调整模块用于：调整算子的最大并行度的计算方式，以增大算子的最大并行度。

[0114] 在示例性的实施例中，数据恢复装置700还包括生成模块(图中未示出)，其中生成模块用于：在满足预设条件的情况下，根据预定规则生成算子的ID，预定规则包括不将上游算子与下游算子chain在一起。

[0115] 在示例性的实施例中，数据恢复装置700还包括关联模块(图中未示出)，其中关联模块用于：在算子的ID发生变化的情况下，获取算子名称和当前作业的有向无环图中算子的第二ID；根据算子名称关联第一ID和第二ID；根据第一ID与第二ID的关联关系，将原检查点的状态数据分配至有向无环图的算子中。

[0116] 实施例三

[0117] 图10示意性示出了根据本申请实施例三的适于数据恢复方法的计算机设备800的硬件架构图。计算机设备800可以是一种能够按照事先设定或者存储的指令，自动进行数值计算和/或数据处理的设备。例如，可以是机架式服务器、刀片式服务器、塔式服务器或机柜式服务器(包括独立的服务器，或者多个服务器所组成的服务器集群)、网关等。如图10所示，计算机设备800至少包括但不限于：可通过系统总线相互通信链接存储器810、处理器820、网络接口830。其中：

[0118] 存储器810至少包括一种类型的计算机可读存储介质，可读存储介质包括闪存、硬盘、多媒体卡、卡型存储器(例如，SD或DX存储器等)、随机访问存储器(RAM)、静态随机访问存储器(SRAM)、只读存储器(ROM)、电可擦除可编程只读存储器(EEPROM)、可编程只读存储

器(PROM)、磁性存储器、磁盘、光盘等。在一些实施例中，存储器810可以是计算机设备800的内部存储模块，例如该计算机设备800的硬盘或内存。在另一些实施例中，存储器810也可以是计算机设备800的外部存储设备，例如该计算机设备800上配备的插接式硬盘，智能存储卡(Smart Media Card，简称为SMC)，安全数字(Secure Digital，简称为SD)卡，闪存卡(Flash Card)等。当然，存储器810还可以既包括计算机设备800的内部存储模块也包括其外部存储设备。本实施例中，存储器810通常用于存储安装于计算机设备800的操作系统和各类应用软件，例如数据恢复方法的程序代码等。此外，存储器810还可以用于暂时地存储已经输出或者将要输出的各类数据。

[0119] 处理器820在一些实施例中可以是中央处理器(Central Processing Unit，简称为CPU)、控制器、微控制器、微处理器、或其他数据处理芯片。该处理器820通常用于控制计算机设备800的总体操作，例如执行与计算机设备800进行数据交互或者通信相关的控制和处理等。本实施例中，处理器820用于运行存储器810中存储的程序代码或者处理数据。

[0120] 网络接口830可包括无线网络接口或有线网络接口，该网络接口830通常用于在计算机设备800与其他计算机设备之间建立通信链接。例如，网络接口830用于通过网络将计算机设备800与外部终端相连，在计算机设备800与外部终端之间的建立数据传输通道和通信链接等。网络可以是企业内部网(Intranet)、互联网(Internet)、全球移动通信系统(Global System of Mobile communication，简称为GSM)、宽带码分多址(Wideband Code Division Multiple Access，简称为WCDMA)、4G网络、5G网络、蓝牙(Bluetooth)、Wi-Fi等无线或有线网络。

[0121] 需要指出的是，图10仅示出了具有部件810-830的计算机设备，但是应理解的是，并不要求实施所有示出的部件，可以替代的实施更多或者更少的部件。

[0122] 在本实施例中，存储于存储器810中的数据恢复方法还可以被分割为一个或者多个程序模块，并由一个或多个处理器(本实施例为处理器820)所执行，以完成本申请实施例。

[0123] 实施例四

[0124] 本申请实施例还提供一种计算机可读存储介质，计算机可读存储介质其上存储有计算机程序，计算机程序被处理器执行时实现实施例中的数据恢复方法的步骤。

[0125] 本实施例中，计算机可读存储介质包括闪存、硬盘、多媒体卡、卡型存储器(例如，SD或DX存储器等)、随机访问存储器(RAM)、静态随机访问存储器(SRAM)、只读存储器(ROM)、电可擦除可编程只读存储器(EEPROM)、可编程只读存储器(PROM)、磁性存储器、磁盘、光盘等。在一些实施例中，计算机可读存储介质可以是计算机设备的内部存储单元，例如该计算机设备的硬盘或内存。在另一些实施例中，计算机可读存储介质也可以是计算机设备的外部存储设备，例如该计算机设备上配备的插接式硬盘，智能存储卡(Smart Media Card，简称为SMC)，安全数字(Secure Digital，简称为SD)卡，闪存卡(Flash Card)等。当然，计算机可读存储介质还可以既包括计算机设备的内部存储单元也包括其外部存储设备。本实施例中，计算机可读存储介质通常用于存储安装于计算机设备的操作系统和各类应用软件，例如实施例中数据恢复方法的程序代码等。此外，计算机可读存储介质还可以用于暂时地存储已经输出或者将要输出的各类数据。

[0126] 显然，本领域的技术人员应该明白，上述的本申请实施例的各模块或各步骤可以

用通用的计算装置来实现，它们可以集中在单个的计算装置上，或者分布在多个计算装置所组成的网络上，可选地，它们可以用计算装置可执行的程序代码来实现，从而，可以将它们存储在存储装置中由计算装置来执行，并且在某些情况下，可以以不同于此处的顺序执行所示出或描述的步骤，或者将它们分别制作成各个集成电路模块，或者将它们中的多个模块或步骤制作成单个集成电路模块来实现。这样，本申请实施例不限制于任何特定的硬件和软件结合。

[0127] 以上仅为本申请的优选实施例，并非因此限制本申请的专利范围，凡是利用本申请说明书及附图内容所作的等效结构或等效流程变换，或直接或间接运用在其他相关的技术领域，均同理包括在本申请的专利保护范围内。

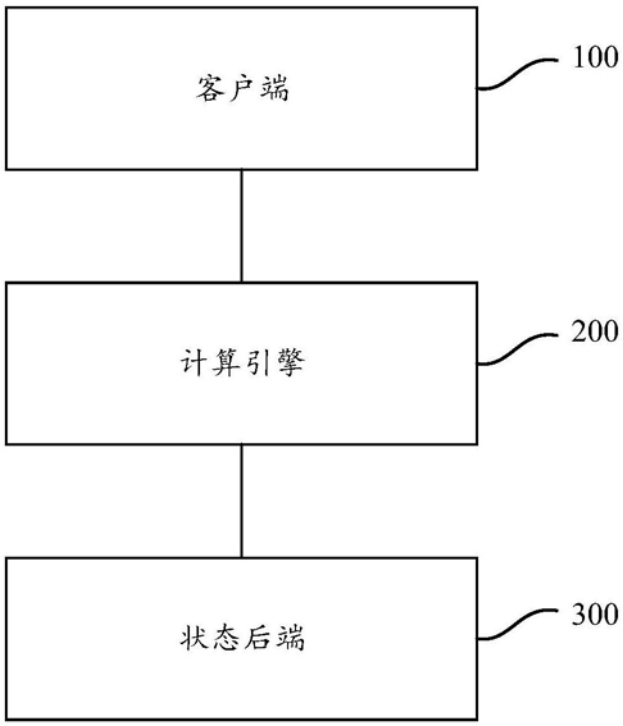



图1


广告
AI
数仓
其它业务
平台层（Saber）
API
SQL
Flink Runtime
HA
Checkpoint
DAG管理
调度/资源管理
网络
外部系统交互
HDFS
Zookeeper
YARN
K8S

图2

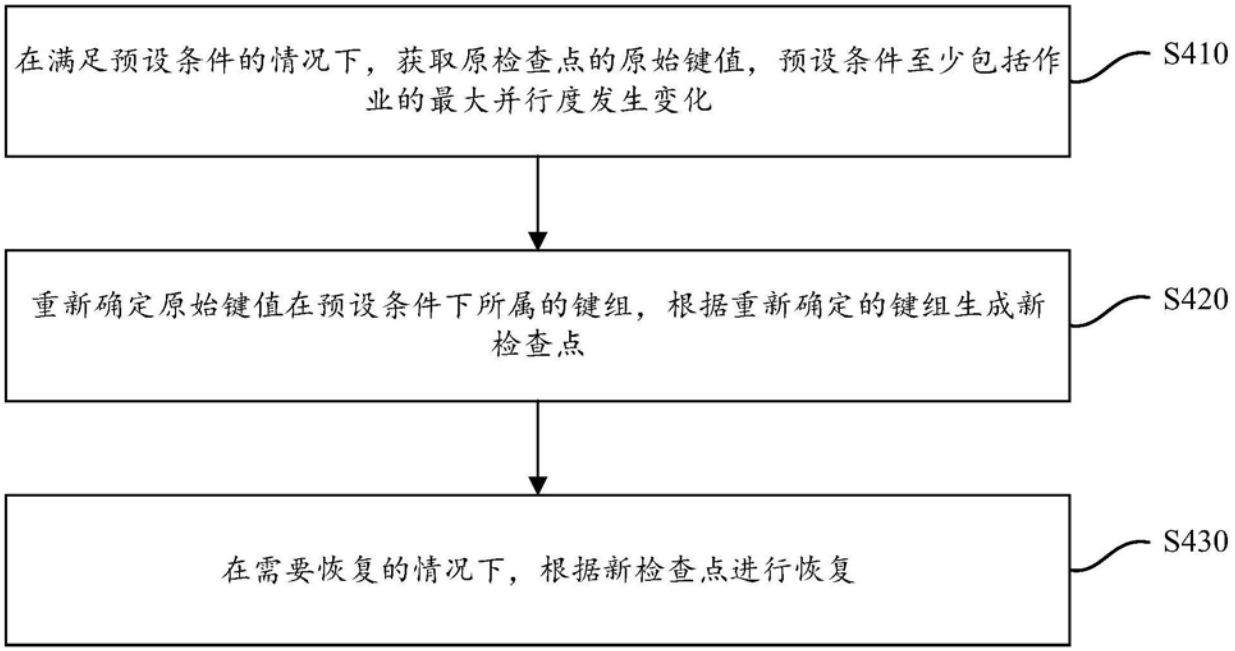

在满足预设条件的情况下，获取原检查点的原始键值，预设条件至少包括作业的最大并行度发生变化
S410
重新确定原始键值在预设条件下所属的键组，根据重新确定的键组生成新检查点
S420
在需要恢复的情况下，根据新检查点进行恢复
S430

图3

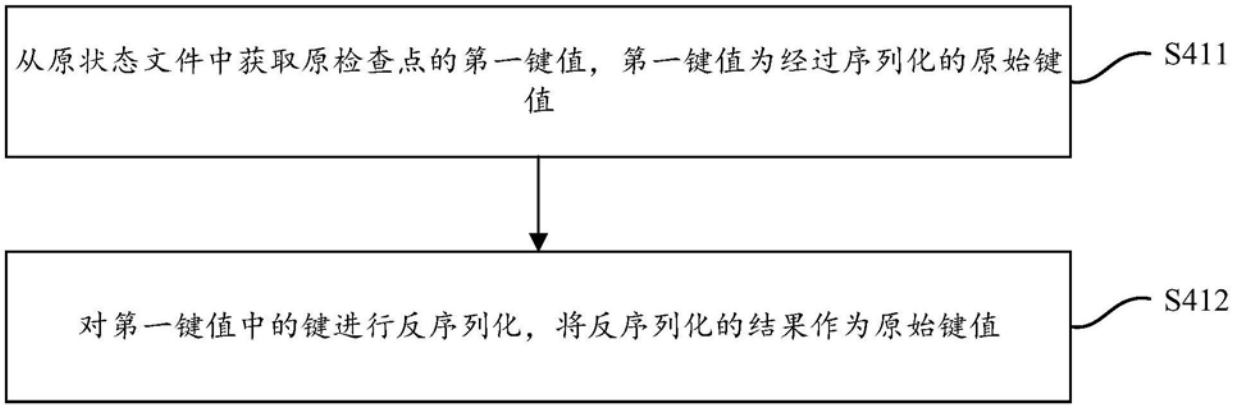

从原状态文件中获取原检查点的第一键值，第一键值为经过序列化的原始键值
S411
对第一键值中的键进行反序列化，将反序列化的结果作为原始键值
S412

图4

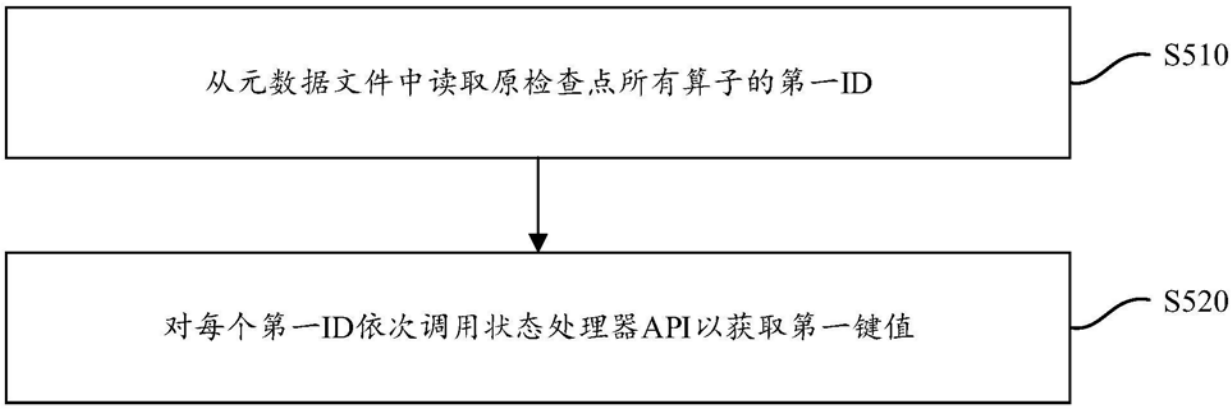

从元数据文件中读取原检查点所有算子的第一ID
S510
对每个第一ID依次调用状态处理器API以获取第一键值
S520

图5

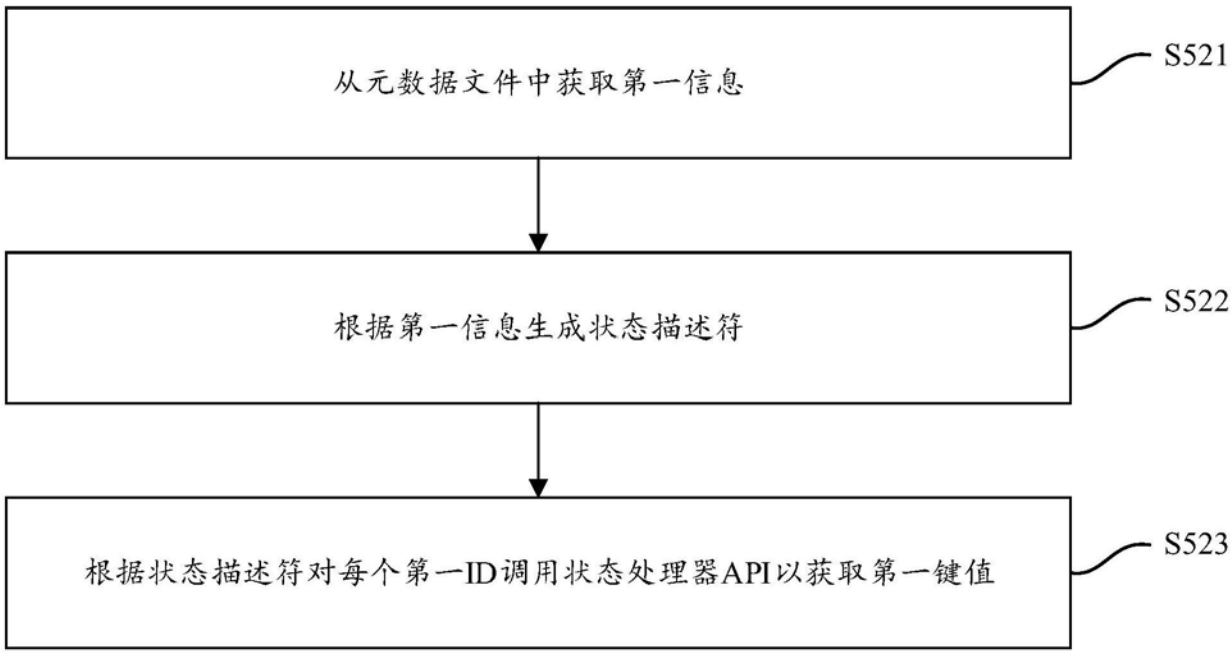

从元数据文件中获取第一信息
S521
根据第一信息生成状态描述符
S522
根据状态描述符对每个第一ID调用状态处理器API以获取第一键值
S523

图6

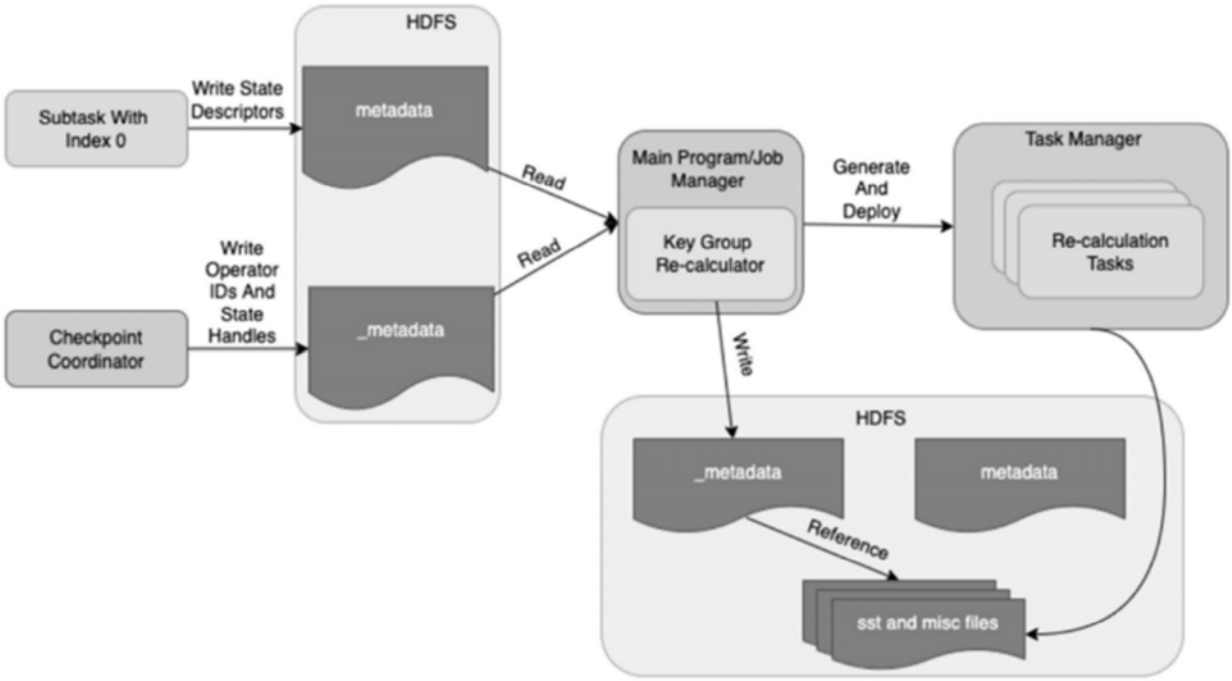

HDFS
Subtask With Index 0
Write State Descriptors
metadata
Read
Main Program/Job Manager
Key Group Re-calculator
Generate And Deploy
Task Manager
Re-calculation Tasks
Write Operator IDs And State Handles
Checkpoint Coordinator
_metadata
Read
Write
HDFS
_metadata
metadata
Reference
sst and misc files

图7

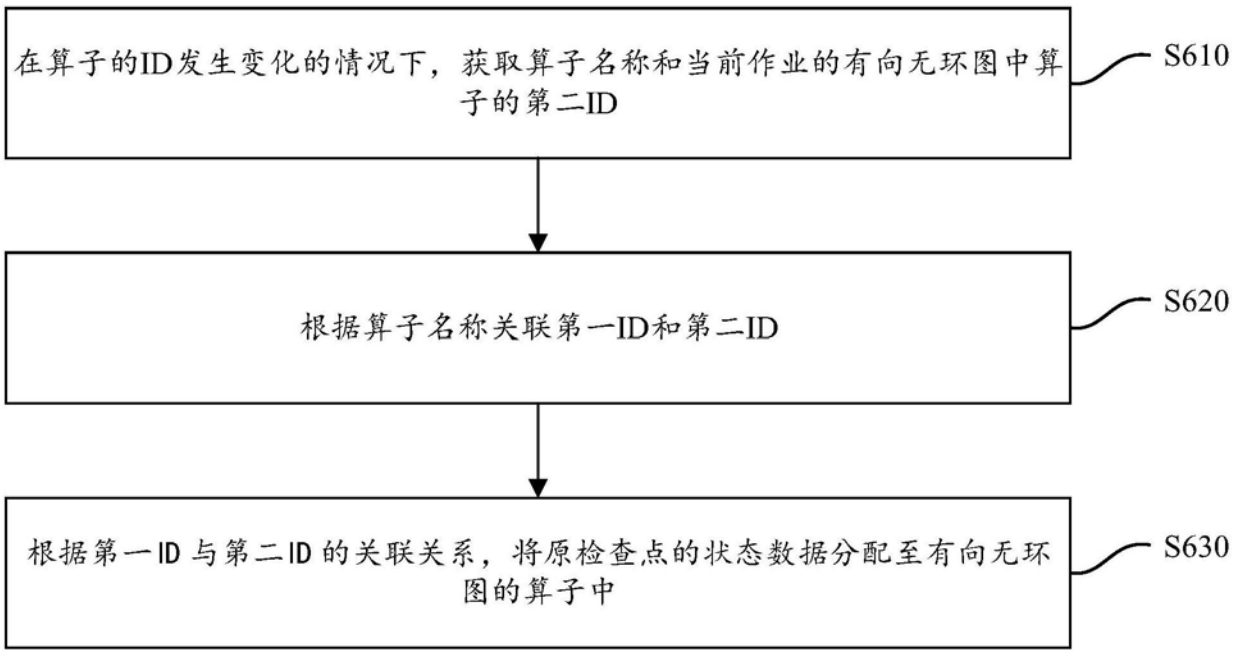

在算子的ID发生变化的情况下，获取算子名称和当前作业的有向无环图中算子的第二ID
S610
根据算子名称关联第一ID和第二ID
S620
根据第一ID与第二ID的关联关系，将原检查点的状态数据分配至有向无环图的算子中
S630

图8

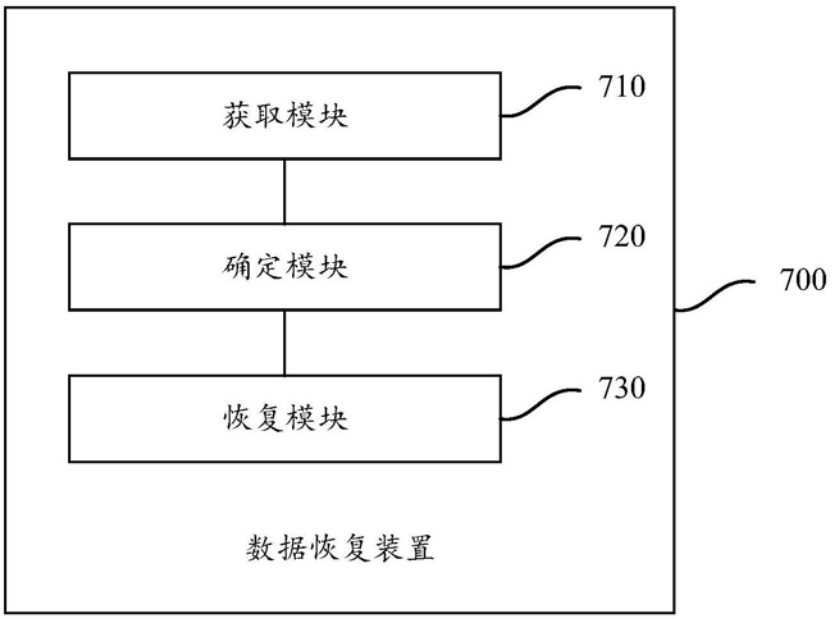

获取模块
710
确定模块
720
700
恢复模块
730
数据恢复装置

图9

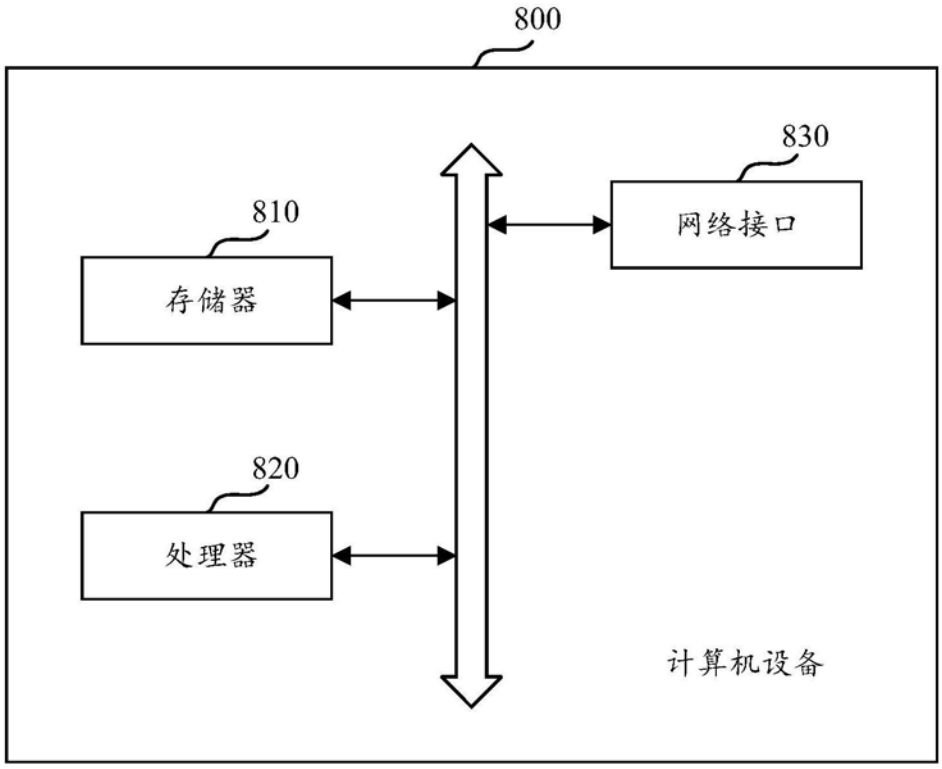

800
830
网络接口
810
存储器
820
处理器
计算机设备

图10