(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 115509635 A
(43)申请公布日 2022.12.23

(21)申请号 202211231336.X

(22)申请日 2022.10.09

(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 胡云海 张杨 王丁

(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 9/445(2018.01)
G06F 9/50(2006.01)

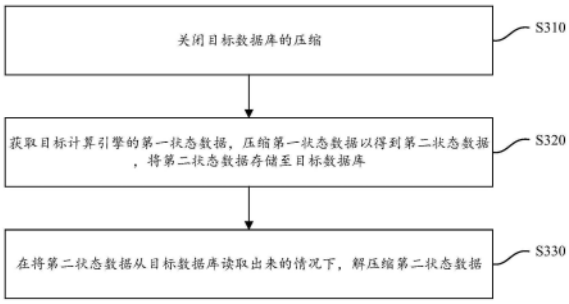
权利要求书1页 说明书8页 附图5页

(54)发明名称

数据处理方法及装置

(57)摘要

本申请实施例提供一种数据处理方法，所述方法包括：关闭目标数据库的压缩；获取目标计算引擎的第一状态数据，压缩所述第一状态数据以得到第二状态数据，将所述第二状态数据存储至所述目标数据库；在将所述第二状态数据从所述目标数据库读取出来的情况下，解压缩所述第二状态数据。本申请实施例提供的数据处理方法，可以减少较大State存储至RocksDB时占用的计算资源，优化Flink的性能。



CN 115509635 A

1.一种数据处理方法,其特征在于,包括:

关闭目标数据库的压缩;

获取目标计算引擎的第一状态数据,压缩所述第一状态数据以得到第二状态数据,将所述第二状态数据存储至所述目标数据库;

在将所述第二状态数据从所述目标数据库读取出来的情况下,解压缩所述第二状态数据。

2.根据权利要求1所述的数据处理方法,其特征在于,所述目标计算引擎为Flink,所述目标数据库为RocksDB。

3.根据权利要求2所述的数据处理方法,其特征在于,所述压缩所述第一状态数据以得到第二状态数据,包括:

在所述Flink的读写层压缩所述第一状态数据,以得到所述第二状态数据。

4.根据权利要求3所述的数据处理方法,其特征在于,所述第一状态数据包括Map状态数据、Value状态数据、递减状态数据和累加状态数据。

5.根据权利要求1-4任一项所述的数据处理方法,其特征在于,所述目标数据库包括切换开关,所述方法还包括:

在所述切换开关打开的情况下,执行所述关闭目标数据库的压缩及之后的步骤。

6.根据权利要求5所述的数据处理方法,其特征在于,还包括:

在所述切换开关关闭的情况下,打开所述目标数据库的压缩,并将所述第一状态数据直接存储至所述目标数据库。

7.一种数据处理装置,其特征在于,包括:

关闭模块,用于关闭目标数据库的压缩;

压缩模块,用于获取目标计算引擎的第一状态数据,压缩所述第一状态数据以得到第二状态数据,将所述第二状态数据存储至所述目标数据库;

解压模块,用于在将所述第二状态数据从所述目标数据库读取出来的情况下,解压缩所述第二状态数据。

8.根据权利要求7所述的数据处理装置,其特征在于,所述目标计算引擎为Flink,所述目标数据库为RocksDB。

9.一种计算机设备,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6中任一项所述的数据处理方法的步骤。

10.一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6中任一项所述的数据处理方法的步骤。

数据处理方法及装置

技术领域

[0001] 本申请涉及计算机技术领域,特别涉及一种数据处理方法、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State(状态数据)存储至RocksDB,从而方便在发生异常时可以进行自我恢复,其中,RocksDB是用于键值数据的高性能嵌入式数据库。

[0003] 对于Flink执行的一些任务(如线上模型特征任务)来说,State会比较大,例如达到500G,有的甚至接达10TB,这些State存储至RocksDB时,由于RocksDB中会存在compact过程的压缩与解压缩,也有block(块)加载到缓存的解压缩,且其中compact过程有明显的读写放大效应,因此会占用过多的计算资源,进而影响Flink的性能。

发明内容

[0004] 本申请的目的在于提供一种数据处理方法、装置、计算机设备及存储介质,用于解决目前较大State存储时会占用过多的计算资源,影响Flink的性能的技术问题。

[0005] 本申请实施例的一个方面提供了一种数据处理方法,包括:关闭目标数据库的压缩;获取目标计算引擎的第一状态数据,压缩所述第一状态数据以得到第二状态数据,将所述第二状态数据存储至所述目标数据库;在将所述第二状态数据从所述目标数据库读取出来的情况下,解压缩所述第二状态数据。

[0006] 可选地,所述目标计算引擎为Flink,所述目标数据库为RocksDB。

[0007] 可选地,所述压缩所述第一状态数据以得到第二状态数据,包括:在所述Flink的读写层压缩所述第一状态数据,以得到所述第二状态数据。

[0008] 可选地,所述第一状态数据包括Map状态数据、Value状态数据、递减状态数据和累加状态数据。

[0009] 可选地,所述目标数据库包括切换开关,所述方法还包括:在所述切换开关打开的情况下,执行所述关闭目标数据库的压缩及之后的步骤。

[0010] 可选地,所述方法还包括:在所述切换开关关闭的情况下,打开所述目标数据库的压缩,并将所述第一状态数据直接存储至所述目标数据库。

[0011] 本申请实施例的一个方面又提供了一种数据处理装置,包括:关闭模块,用于关闭目标数据库的压缩;压缩模块,用于获取目标计算引擎的第一状态数据,压缩所述第一状态数据以得到第二状态数据,将所述第二状态数据存储至所述目标数据库;解压模块,用于在将所述第二状态数据从所述目标数据库读取出来的情况下,解压缩所述第二状态数据。

[0012] 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处理方法的步骤。

[0013] 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处理方法的步骤。

[0014] 本申请实施例提供的数据处理方法、装置、计算机设备及存储介质，包括以下优点：

[0015] 通过关闭目标数据库的压缩，获取目标计算引擎的第一状态数据，压缩第一状态数据以得到第二状态数据，将第二状态数据存储至目标数据库；在将第二状态数据从目标数据库读取出来的情况下，解压缩第二状态数据。由于关闭目标数据库(如RocksDB)的压缩，可以减少目标数据库在压缩和解压缩的过程由于读写放大效应对CPU等计算资源的占用，从而减少对Flink性能的影响，优化了Flink的性能；同时，在将写入目标数据库的状态数据先压缩后再写入，可以避免状态数据过多在目标数据库中膨胀很多倍，避免造成的过多的存储空间占用；另外，在将状态数据从目标数据库中读取出来之后进行解压缩，可以还原相应的状态数据，从而方便状态数据的使用。

附图说明

[0016] 图1示意性示出了本申请实施例的环境架构图；

[0017] 图2示意性示出了本申请实施例一的数据处理方法的流程图；

[0018] 图3为Flink及RocksDB原始运作模式的示意图；

[0019] 图4为Flink及RocksDB采用本申请实施例的数据处理方法的运作模式的示意图；

[0020] 图5为采用本申请实施例的数据处理方法前后计算资源占用的对比示例图；

[0021] 图6为采用本申请实施例的数据处理方法前后文件大小的对比示例图；

[0022] 图7为采用本申请实施例的数据处理方法前后RocksDB读写耗时的对比示例图；

[0023] 图8示意性示出了本申请实施例二的数据处理装置的框图；

[0024] 图9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

[0025] 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。

[0026] 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。

[0027] 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。

[0028] 下面为本申请涉及的术语解释：

[0029] 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流引擎，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。

[0030] 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，可以利用许多CPU内核，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，为基于日志结构的合并树(LSM树)数据结构。

[0031] savepoint，是在数据库事务处理中实现“子事务”(subtransaction)，也称为嵌套事务(nested transaction)的方法，事务可以回滚到savepoint而不影响savepoint创建前的变化，不需要放弃整个事务。

[0032] 图1示意性示出了本申请实施例的环境架构图，如图所示：

[0033] 目标计算引擎100与目标数据库200相连接。目标计算引擎100在运行过程中产生状态数据，这些状态数据会由目标计算引擎100写入至目标数据库200中。在需要读取时，目标计算引擎100再从目标数据库200中读取状态数据。不同于相关技术的是，目标计算引擎100在将状态数据写入目标数据库200之前，先关闭目标数据库200自身的压缩，并将状态数据进行压缩，然后再将压缩后的数据写入至目标数据库200；在读取时，目标计算引擎100将压缩后的状态数据从目标数据库200读取出来之后，对状态数据进行解压缩，从而得到相应的状态数据。

[0034] 目标计算引擎100可以包括但不限于运行中产生状态数据的计算引擎，例如Flink，其中状态数据指计算过程状态的数据。目标数据库200可以包括但不限于自身带有压缩的数据库，例如LSM-Tree(LSM树)数据库，具体可以是RocksDB或Hbase等。

[0035] 相关技术中，Flink运行中产生的一些较大的状态数据在存储至RocksDB时，会占用Flink及RocksDB过多的计算资源，进而影响Flink的性能。

[0036] 本申请实施例的数据处理方案，可以使Flink在存储较大的状态数据至RocksDB时，减少对计算资源的占用和对Flink性能的影响。

[0037] 以下将通过若干个实施例介绍数据处理方案，为便于理解，下面将以图1中的目标计算引擎100为执行主体进行示例性描述。

[0038] 实施例一

[0039] 图2示意性示出了本申请实施例一的数据处理方法的流程图，包括步骤S310～步骤S330，具体说明如下：

[0040] 步骤S310，关闭目标数据库的压缩。

[0041] 实际应用中，图1中的目标计算引擎100可以包括客户端，可以由用户在客户端通过更改目标数据库200的配置来关闭目标数据库200的压缩。在这种情况下，步骤S310具体为：目标计算引擎100根据其客户端用户的输入指令，关闭目标数据库200的压缩。

[0042] 可选地，也可以在目标计算引擎100中设置相应的触发条件来触发其关闭目标数据库200的压缩，并执行后续的步骤。例如，可以设置一个预设阈值，在确定第一状态数据的大小达到或超过该预设阈值的情况下，执行步骤S310及之后的步骤，从而实现自动触发的效果。具体的触发条件可以根据实际需要进行设置，此处不做具体限制。

[0043] 步骤S320，获取目标计算引擎的第一状态数据，压缩第一状态数据以得到第二状

态数据,将第二状态数据存储至目标数据库。

[0044] 即,目标计算引擎100在将状态数据存储至目标数据库200之前,先将状态数据(第一状态数据)压缩,再将压缩后的状态数据(第二状态数据)存储至目标数据库200。目标计算引擎100在将第一状态数据压缩为第二状态数据时,具体采用的压缩算法可以根据实际需要进行选择,此处不做具体限制。相应地,步骤S330中,对第二状态数据进行解压缩的方法与压缩时采用的压缩算法对应,从而可以还原出相应的状态数据(即第一状态数据)。

[0045] 在示例性的实施例中,目标计算引擎为Flink,目标数据库为RocksDB。第一状态数据可以包括但不限于Map状态数据、Value状态数据、递减(Reducing)状态数据和累加(Aggregating)状态数据。

[0046] 在示例性的实施例中,压缩第一状态数据以得到第二状态数据,可以包括:在Flink的读写层压缩第一状态数据,以得到第二状态数据。其中,Flink的读写层具体是指Flink的backend(后端),即在Flink的backend中实现对状态数据的压缩。

[0047] 步骤S330,在将第二状态数据从目标数据库读取出来的情况下,解压缩第二状态数据。

[0048] 由于第二状态数据是经过压缩的数据,因此目标计算引擎100在将第二状态数据从目标数据库200读取出来后,解压缩第二状态数据,可以得到第一状态数据,也即压缩前的状态数据,从而可以正常使用相应的状态数据。

[0049] 应当理解的是,关闭目标数据库200(如RocksDB)的压缩,是为了减少其在compact等过程的读写放大效应,从而减少将较大的状态数据存储至目标数据库200时计算资源的占用。在可选的实施例中,可以直接关闭目标数据库200的压缩,同时直接将状态数据不经压缩写入目标数据库200,但由于状态数据没有经过压缩的过程,因此相比原来使用目标数据库200压缩的方案状态数据会在目标数据库200中膨胀很多倍,占用较大的存储空间。本申请实施例中,在关闭目标数据库200自身的压缩的同时,将状态数据压缩后再写入目标数据库200,在读取出来之后再解压缩状态数据,可以减少将状态数据存储至目标数据库200时对计算资源的占用,也能避免占用目标数据库200较多的存储空间。另外,由于RocksDB是内置于Flink中的,因此减少状态数据存储至目标数据库200时对计算资源的占用,可以优化Flink的性能,提高Flink处理业务的效率,避免Flink在进行checkpoint或处理其它业务时发生抖动或堵塞等问题。

[0050] 请参考图3,其为Flink及RocksDB原始运作的模式。如图所示,Flink从外部获取数据(values)进行处理,运行过程中会产生MapState(Map状态数据)、ValueState(Value状态数据)、ReducingState(递减状态数据)和AggregatingState(累加状态数据),这些状态数据直接写入RocksDB。在状态数据较大时,容易因RocksDB中compact等过程的读写放大而占用较多的计算资源,从而影响Flink的性能。

[0051] 请参考图4,其为Flink及RocksDB采用本申请实施例的数据处理方法的运作模式。如图所示,关闭RocksDB自身的压缩,将Flink运行过程的状态数据先经过压缩(对应图中的compress)再写入RocksDB(对应图中的put),从RocksDB读出来后(对应图中的get),进行解压缩(对应图中的decompress)。由于RocksDB自身的压缩已关闭,因此其自身的压缩所带来的读写放大效应会减少,从而减少对计算资源的占用;同时,由于状态数据是经过压缩后写入,因此相比原来使用目标数据库200压缩的方案,不会造成状态数据在RocksDB中会膨胀

很多倍的结果,减少对存储空间的占用。

[0052] 请参考图5,其为采用本申请实施例的数据处理方法前后计算资源占用的对比图。从图5中可以看出,采用本申请实施例的数据处理方法之前,Flink中的TaskManager(任务管理器)使用的CPU核数的平均值(avg)为76;而采用本申请实施例的数据处理方法后,TaskManager使用的CPU核数的平均值(avg)为55,占用的计算资源有较大程度的下降。并且,从图中还可以看出,状态数据较大时,两者占用的计算资源相差较大,因此在状态数据较大时采用本申请实施例的数据处理方法优化效果会比较明显;而在状态数据较小时,两者占用的计算资源相差不大,因此在一些状态数据较小的情况,由于优化效果不够明显,也可以直接使用原来的方案。

[0053] 请参考图6,其为采用本申请实施例的数据处理方法前后SST文件大小的对比图。从图中可以看出,采用本申请实施例的数据处理方法前后的压缩大小相差不大,在部分情况下,采用本申请的实施例的数据处理方法的压缩效果要好于采用前的压缩效果。

[0054] 请参考图7,其为采用本申请实施例的数据处理方法前后RocksDB读写耗时的对比图。图7中右下角从左往右的第三列数据为RocksDB读写耗时的平均值数据,其中第一行数据和第四行数据分别对应采用前后的数据。从图中可以看出,采用本申请实施例的数据处理方法之前,RocksDB读写耗时的平均值为70.209ms;而采用本申请实施例的数据处理方法后,RocksDB读写耗时的平均值为43.228ms,读写速度有较大的提升。也即是说,将状态数据压缩之后再写入RocksDB或从RocksDB中读出,可以提高相应的读写效率。

[0055] 本申请实施例的数据处理方法,通过关闭目标数据库的压缩,获取目标计算引擎的第一状态数据,压缩第一状态数据以得到第二状态数据,将第二状态数据存储至目标数据库;在将第二状态数据从目标数据库读取出来的情况下,解压缩第二状态数据。由于关闭目标数据库(如RocksDB)的压缩,可以减少目标数据库自身的压缩所带来的读写放大效应对CPU等计算资源的占用,从而减少对Flink性能的影响,优化了Flink的性能;同时,在将写入目标数据库的状态数据先压缩后再写入,可以避免状态数据过多在目标数据库中膨胀很多倍,避免造成的过多的存储空间占用;另外,在将状态数据从目标数据库中读取出来之后进行解压缩,可以还原相应的状态数据,从而方便状态数据的使用。

[0056] 在示例性的实施例中,目标数据库200包括切换开关,数据处理方法还可以包括:在切换开关打开的情况下,执行关闭目标数据库的压缩(即步骤S310)及之后的步骤。

[0057] 具体地,可以在目标数据库200中增加相应的切换开关,在切换开关打开时,执行图2对应的方法步骤。也即,在切换开关打开时,实施关闭目标数据库200的压缩,同时将状态数据压缩后写入目标数据库200以及解压缩从目标数据库200读取出来的状态数据的方案。可选地,目标数据库200的切换开关可以通过目标计算引擎100进行控制,例如通过目标计算引擎100的客户端进行相应的控制。目标计算引擎100在运行时对切换开关的状态进行确认,若切换处于打开的情况,则执行图2对应的方法步骤。

[0058] 实际应用中,由于目标数据库200(如RocksDB)会存在savepoint,savepoint记录的数据在切换后(即采用图2对应的方案)会发生数据结构的变化(从未压缩变为压缩)而使得记录的数据不可用,因此在实施切换时,可以暂时放弃savepoint,以顺利地切换至图2对应的方案,避免因记录的数据不可用而导致的错误。

[0059] 在示例性的实施例中,数据处理方法还可以包括:在切换开关关闭的情况下,打开

目标数据库200的压缩，并将第一状态数据直接存储至目标数据库200。

[0060] 可选地，在切换开关关闭的情况下，若目标数据库200的压缩已经打开，则维持目标数据200的压缩打开的状态。

[0061] 具体地，在切换开关关闭的情况下，打开目标数据库200的压缩，且目标计算引擎100将状态数据不压缩而直接存储至目标数据库200。相应地，由于状态数据是不经过压缩而直接存储至目标数据库200的，因此从目标数据库200读取出来也不需要进行解压缩。

[0062] 上述两个实施例中，通过切换开关来切换状态数据压缩的方案，可以使目标计算引擎100和目标数据库200实现状态数据压缩方案的一键切换，从而使用户可以在状态数据较大的情况下切换至关闭目标数据库200的压缩、在写入之前压缩状态数据的方案，以减少对计算资源占用，优化目标计算引擎100的性能；也能在状态数据较小的情况下切换至打开目标数据库200的压、直接写入状态数据的方案，提高方案切换的灵活性。

[0063] 实施例二

[0064] 图8示意性示出了根据本申请实施例二的数据处理装置400的框图，该数据处理装置400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。

[0065] 如图8所示，该数据处理装置400可以包括关闭模块410、压缩模块420和解压模块430。

[0066] 关闭模块410，用于关闭目标数据库的压缩；

[0067] 压缩模块420，用于获取目标计算引擎的第一状态数据，压缩第一状态数据以得到第二状态数据，将第二状态数据存储至目标数据库；

[0068] 解压模块430，用于在将第二状态数据从目标数据库读取出来的情况下，解压缩第二状态数据。

[0069] 在示例性的实施例中，目标计算引擎为Flink，目标数据库为RocksDB。

[0070] 在示例性的实施例中，压缩模块420用于：在Flink的读写层压缩第一状态数据，以得到第二状态数据。

[0071] 在示例性的实施例中，第一状态数据包括Map状态数据、Value状态数据、递减状态数据和累加状态数据。

[0072] 在示例性的实施例中，目标数据库包括切换开关，该数据处理装置400还包括切换模块(图中未示出)，其中，切换模块用于：在切换开关打开的情况下，执行关闭目标数据库的压缩及之后的步骤。

[0073] 在示例性的实施例中，切换模块还用于：在切换开关关闭的情况下，打开目标数据库的压缩，并将第一状态数据直接存储至目标数据库。

[0074] 实施例三

[0075] 图9示意性示出了根据本申请实施例三的适于数据处理方法的计算机设备500的硬件架构图。计算机设备500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9所示，

计算机设备500至少包括但不限于:可通过系统总线相互通信链接存储器510、处理器520、网络接口530。其中:

[0076] 存储器510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510可以是计算机设备500的内部存储模块,例如该计算机设备500的硬盘或内存。在另一些实施例中,存储器510也可以是计算机设备500的外部存储设备,例如该计算机设备500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510还可以既包括计算机设备500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510通常用于存储安装于计算机设备500的操作系统和各类应用软件,例如数据处理方法的程序代码等。此外,存储器510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。

[0077] 处理器520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520通常用于控制计算机设备500的总体操作,例如执行与计算机设备500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520用于运行存储器510中存储的程序代码或者处理数据。

[0078] 网络接口530可包括无线网络接口或有线网络接口,该网络接口530通常用于在计算机设备500与其他计算机设备之间建立通信链接。例如,网络接口530用于通过网络将计算机设备500与外部终端相连,在计算机设备500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。

[0079] 需要指出的是,图9仅示出了具有部件510-530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。

[0080] 在本实施例中,存储于存储器510中的数据处理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520)所执行,以完成本申请实施例。

[0081] 实施例四

[0082] 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处理方法的步骤。

[0083] 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，例如实施例中数据处理方法的程序代码等。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。

[0084] 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。

[0085] 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。

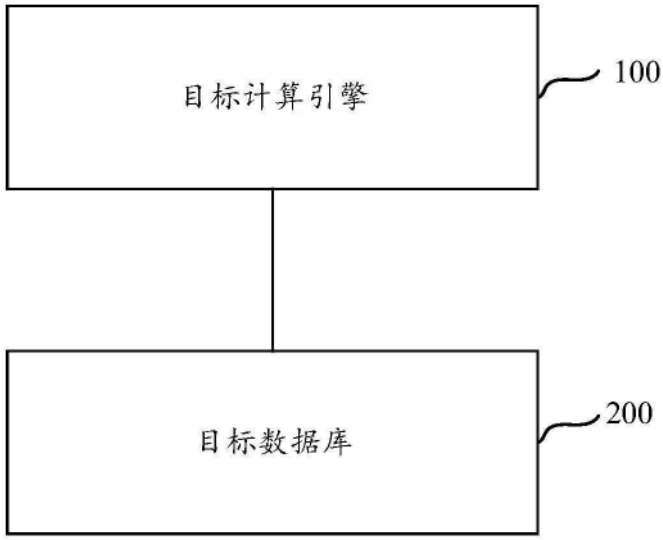

目标计算引擎
100
目标数据库
200

图1

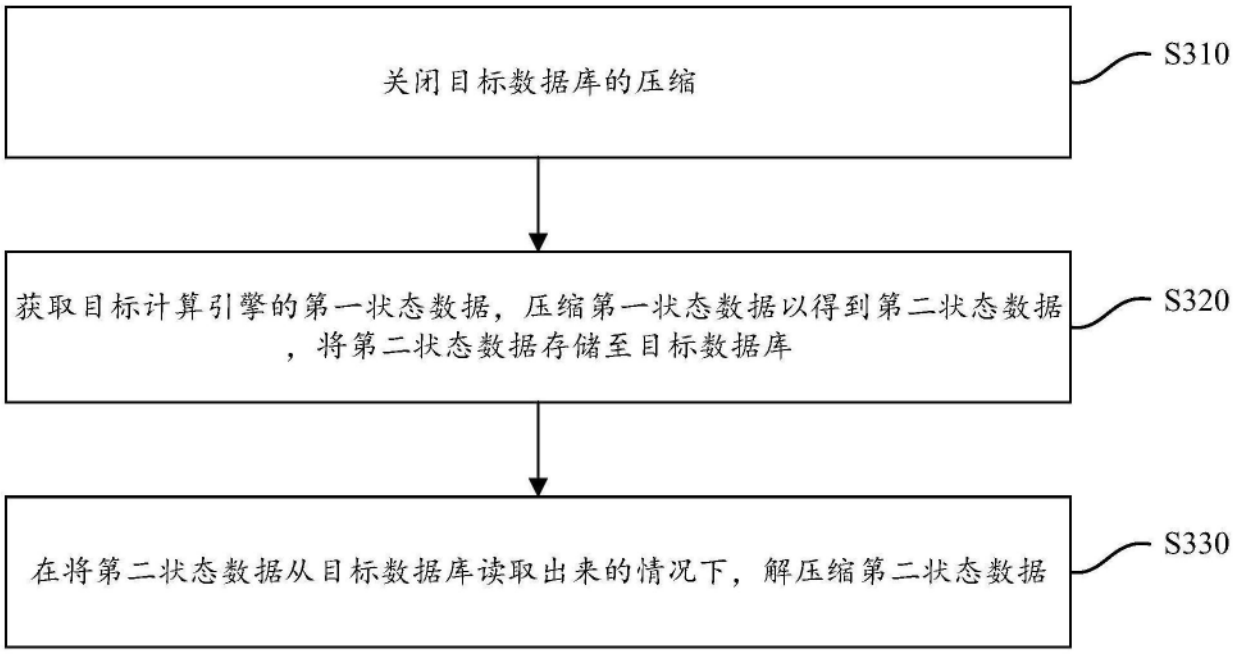

关闭目标数据库的压缩
S310
获取目标计算引擎的第一状态数据，压缩第一状态数据以得到第二状态数据，将第二状态数据存储至目标数据库
S320
在将第二状态数据从目标数据库读取出来的情况下，解压缩第二状态数据
S330

图2

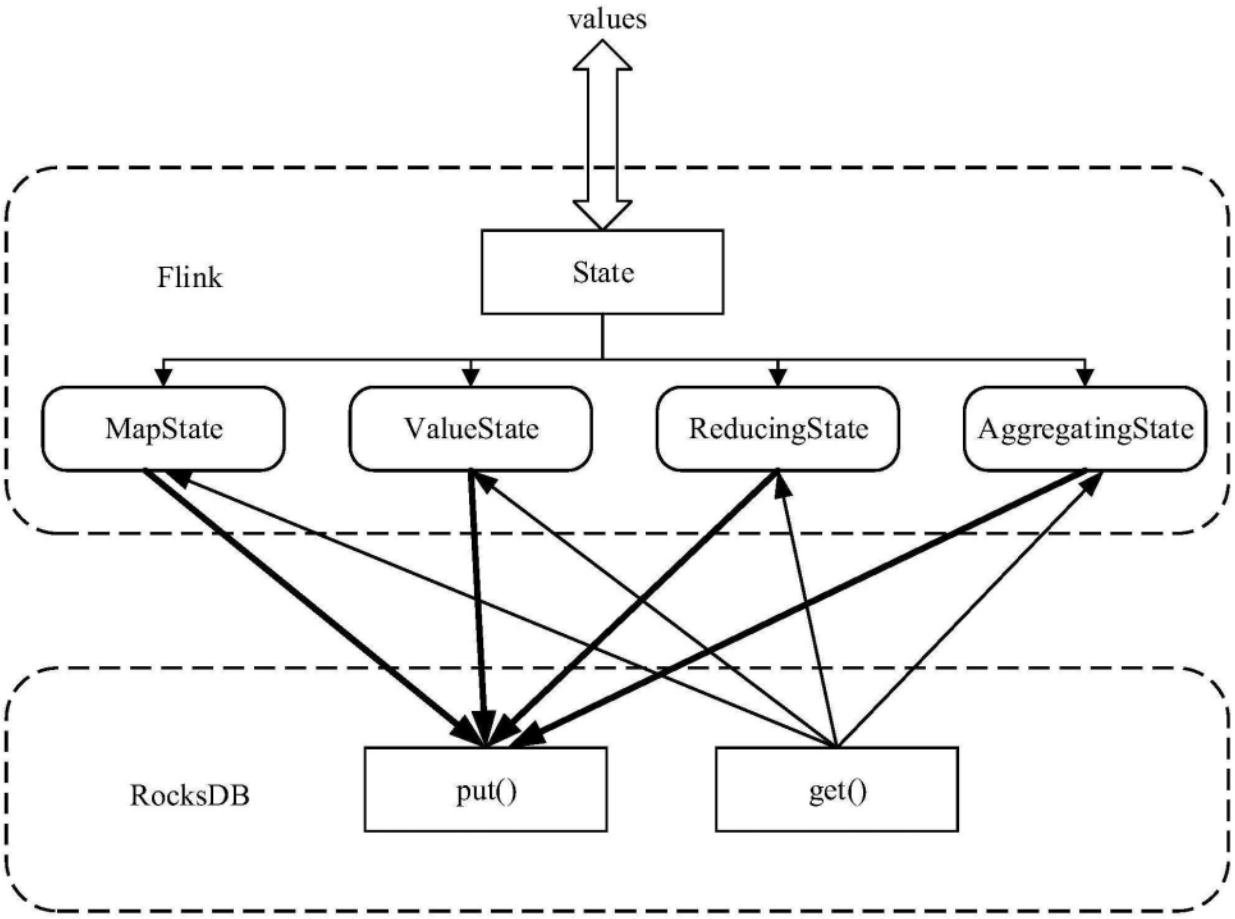

values
Flink
State
MapState
ValueState
ReducingState
AggregatingState
RocksDB
put()
get()

图3

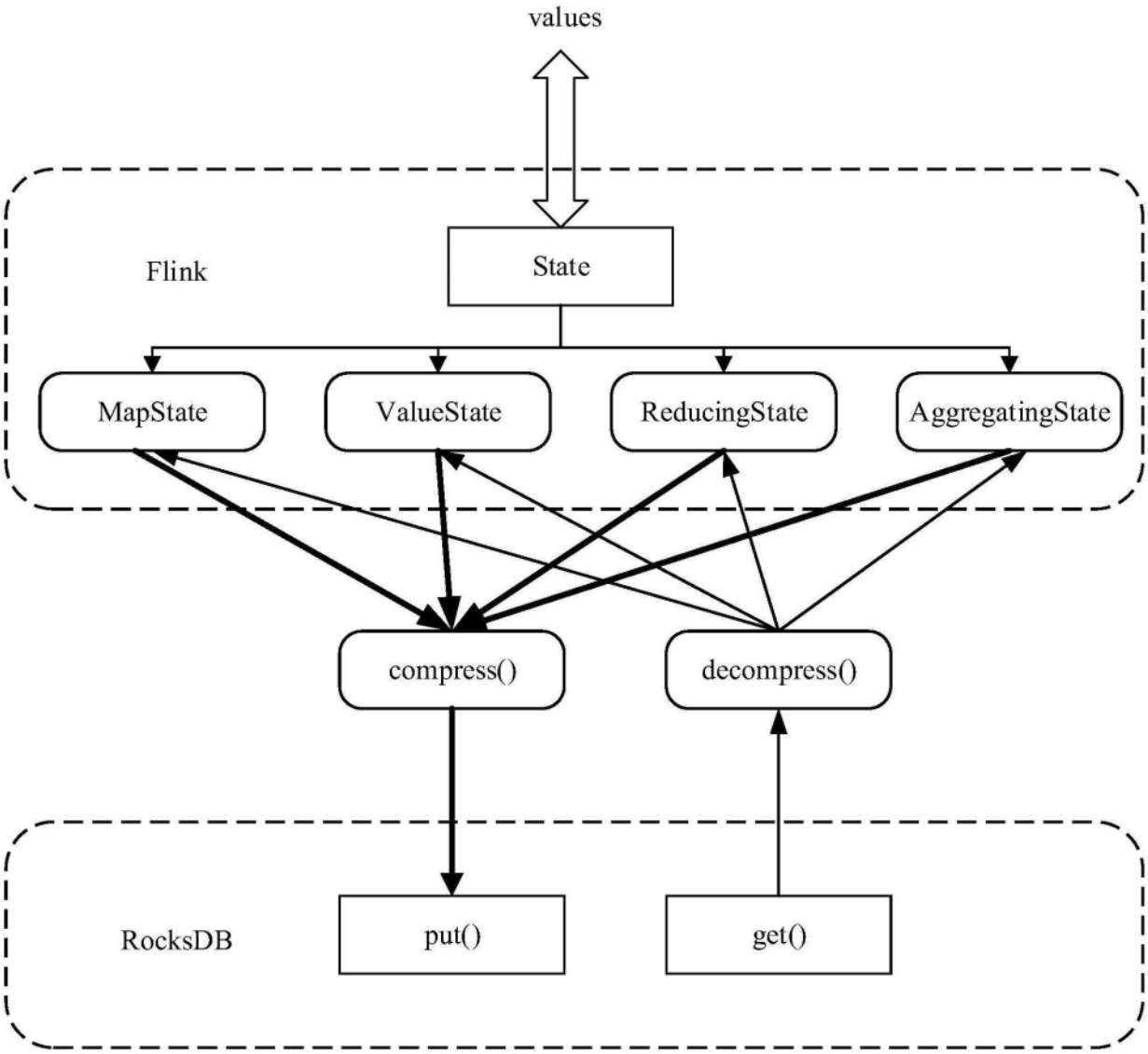

values
Flink
State
MapState
ValueState
ReducingState
AggregatingState
compress()
decompress()
RocksDB
put()
get()

图4


Task Manager CPU使用核数
150
100
50
0
8/19 20:00
8/20 00:00
8/20 04:00
8/20 08:00
8/20 12:00
8/20 16:00
8/20 20:00
8/21 00:00
8/21 04:00
8/21 08:00
8/21 12:00
8/21 16:00
max avg current
yunhai_rocksdb_noCompress:total_cpu_usage 129 76 96
yunhai_rocksdb_compression:total_cpu_usage 120 55 70

图5

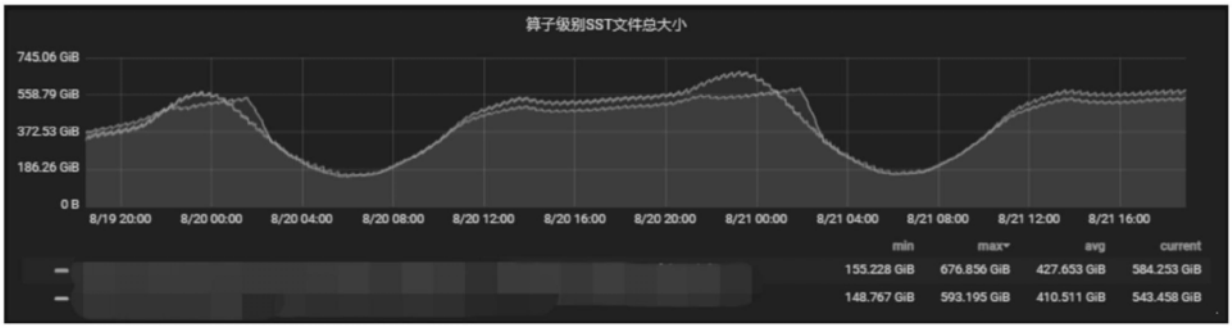

算子级别SST文件总大小
745.06 GiB
558.79 GiB
372.53 GiB
186.26 GiB
0 B
8/19 20:00
8/20 00:00
8/20 04:00
8/20 08:00
8/20 12:00
8/20 16:00
8/20 20:00
8/21 00:00
8/21 04:00
8/21 08:00
8/21 12:00
8/21 16:00
min
max
avg
current
155.228 GiB
676.856 GiB
427.653 GiB
584.253 GiB
148.767 GiB
593.195 GiB
410.511 GiB
543.458 GiB

图6

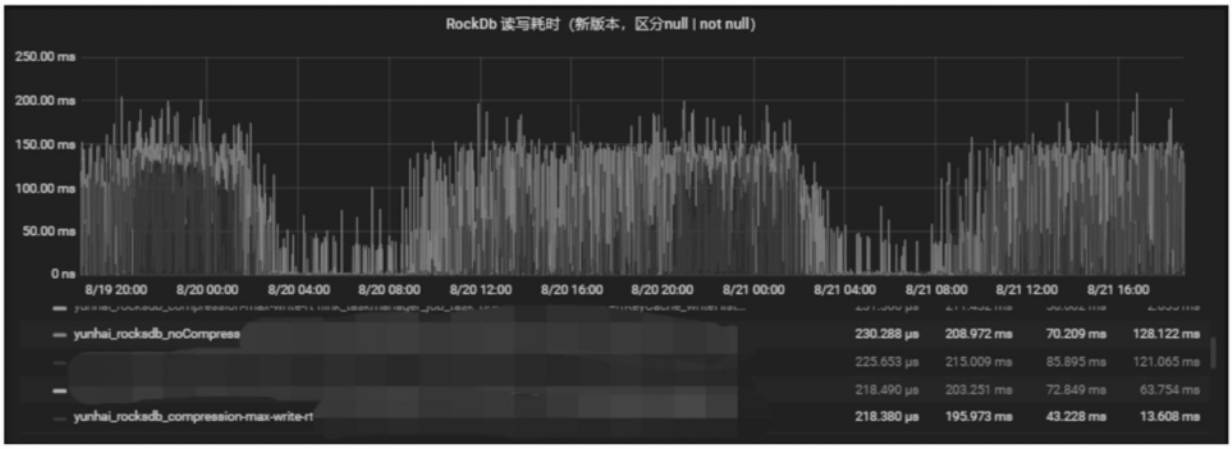

RockDb 读写耗时（新版本，区分null | not null）
250.00 ms
200.00 ms
150.00 ms
100.00 ms
50.00 ms
0 ns
8/19 20:00
8/20 00:00
8/20 04:00
8/20 08:00
8/20 12:00
8/20 16:00
8/20 20:00
8/21 00:00
8/21 04:00
8/21 08:00
8/21 12:00
8/21 16:00
yunhai_rocksdb_noCompress
230.288 µs
208.972 ms
70.209 ms
128.122 ms
225.653 µs
215.009 ms
85.895 ms
121.065 ms
218.490 µs
203.251 ms
72.849 ms
63.754 ms
yunhai_rocksdb_compression-max-write-rt
218.380 µs
195.973 ms
43.228 ms
13.608 ms

图7

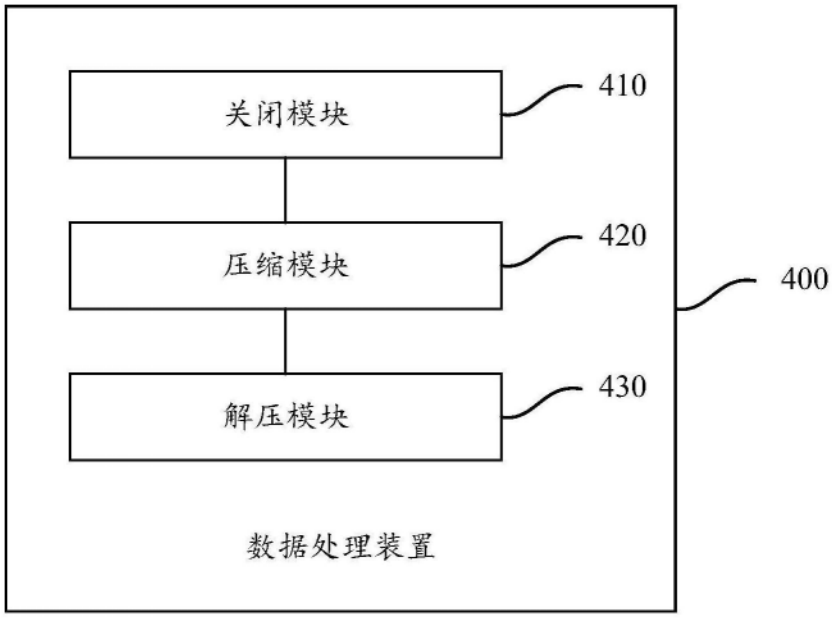

关闭模块
410
压缩模块
420
400
解压模块
430
数据处理装置

图8

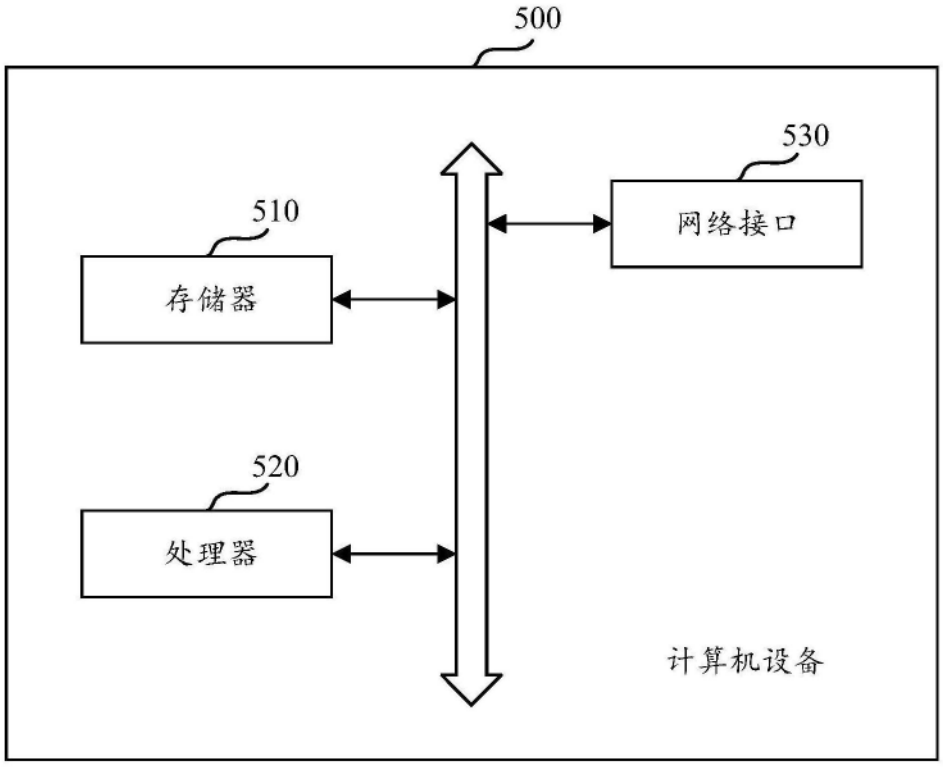


图9