(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 118245044 A
(43)申请公布日 2024.06.25

(21)申请号 202311706616.6

(22)申请日 2023.12.13

(71)申请人 中盈优创资讯科技有限公司
地址 201804 上海市嘉定区曹安公路4811号702室-2

(72)发明人 周朝卫 邱勇

(74)专利代理机构 上海嘉蓝专利代理事务所(普通合伙) 31407
专利代理师 金波

(51)Int.Cl.
*G06F 8/36*(2018.01)
*G06F 8/10*(2018.01)
*G06F 16/25*(2019.01)

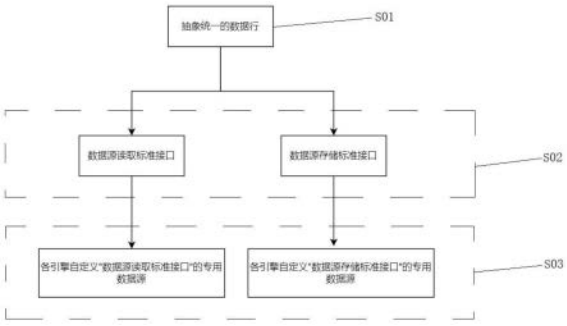
权利要求书4页 说明书15页 附图2页

(54)发明名称

一种解耦计算引擎的异构数据源的接入方法及装置

(57)摘要

本发明公开一种解耦计算引擎的异构数据源的接入方法及装置,其中方法包括:抽象统一的数据行,在数据行类型模型、数据行模型和数据类型方面进行抽象,从而实现各类异构数据源的统一接入和存储,在结构、功能方面具备扩展能力;数据的读取和写入基于插件机制,各类数据源分别实现各自的数据源读取和数据写入的插件,插件的实现规范则通过Java的父级类进行定义;计算引擎的适配,计算引擎通过引擎各自的数据读取和写入的定义接口,定义一种专用的数据源。本发明一种解耦计算引擎的异构数据源的接入方法及装置,使用统一的API来处理数据源的查询和写入操作,从而解耦不同计算引擎间的依赖关系,支持多种计算引擎,大大降低开发和维护的成本。




CN 118245044 A

1.一种解耦计算引擎的异构数据源的接入方法,其特征在于,该方法包括:

S01、抽象统一的数据行,在数据行类型模型、数据行模型和数据类型方面进行抽象,从而实现各类异构数据源的统一接入和存储,在结构、功能方面具备扩展能力;

S02、数据的读取和写入基于插件机制,各类数据源分别实现各自的数据源读取和数据写入的插件,插件的实现规范则通过Java的父级类进行定义;

S03、计算引擎的适配,计算引擎通过引擎各自的数据读取和写入的定义接口,定义一种专用的数据源。

2.根据权利要求1所述的解耦计算引擎的异构数据源的接入方法,其特征在于,所述S01包括:

S011、抽象数据行类型模型描述数据行中的字段、各字段的属性,字段的属性包括:字段的名称、数据类型、是否允许为空值;

S012、数据行模型定义行数据的信息,在数据源的读取和写入过程中,用于交换和传输数据;

S013、抽象字段的统一数据类型,字段的数据类型在数据行类型中表示一个字段的数据类型;建立各个数据源的数据类型和统一数据类型之间的映射关系,使用统一的API实现数据的接入和存储;映射关系的建立是由各个数据源读取和写入插件完成的;在数据源读取时,将数据源的数据类型转换为统一数据类型。

3.根据权利要求2所述的解耦计算引擎的异构数据源的接入方法,其特征在于,所述数据行模型包括:

行数据的类型定义,通过数据行类型模型定义行数据对应的字段信息;

对象数组,存储记录的各个字段值,每个位置对应一个字段;

字段访问索引,通过索引直接访问对象数组内容;

字段名称映射,保证行类型模型定义中的字段名与对象数组位置对应,支持灵活访问;

序列化器,由数据行类型模型与数据行建立关联并负责数据行的序列化与反序列化工作,从而实现数据高效传输;

元数据,记录数据来源、长度、版本辅助描述数据行的信息;

持久化处理器,支持数据行持久化为文件或数据库表等,通过转换数据行类型实现。

4.根据权利要求2所述的解耦计算引擎的异构数据源的接入方法,其特征在于,所述数据类型包括:

基本数据类型包括:整型、字符串、布尔;

复合数据类型:多个基本数据类型组成的数据类型;

数组类型:定义数组的数据类型,数据的元素的数据类型通过基本的数据类型进行定义;

Map映射数据类型:用于定义由key和value之间关系的数据类型,分别定义里key和value的数据类型。

5.根据权利要求1所述的解耦计算引擎的异构数据源的接入方法,其特征在于,所述S02包括:

S021、数据源读取标准接口,利用数据源读取定义接口,限定数据源的元数据和基本行为,数据源读取标准接口包括:数据源读取定义接口、数据源动态分区定义接口与分区数据

读取接口;

数据源读取定义接口负责描述整个数据源,并创建数据源动态分区定义接口和分区数据读取接口;

数据源动态分区定义接口负责不断产出分区对象描述每个分区,分区对象由调度框架分配给不同任务执行子任务;

分区数据读取接口由子任务调用,读取分配给子任务的分区对象的数据,进行真实的数据记录的读取和处理;

S022、数据存储标准接口,接收数据并写入各类数据存储,数据存储标准接口包括:数据源存储定义接口、分区数据写入接口;

数据源存储定义接口负责描述整个数据源存储,创建分区数据写入接口;

分区数据写入接口接收分区对象的数据,每个分区对象由调度框架分配一个任务执行的子任务,子任务调用分区数据写入接口,将分区对象的数据写入目标存储。

6.根据权利要求5所述的解耦计算引擎的异构数据源的接入方法,其特征在于,所述数据源读取定义接口:利用数据源读取定义接口,描述数据源的元数据和基本行为。

7.根据权利要求5所述的解耦计算引擎的异构数据源的接入方法,其特征在于,所述数据源读取定义接口的功能包括:

定义数据源的连接参数:提供连接地址、配置等信息配置数据源;

提供数据读取接口:从数据源读取数据,转换为统一的数据行格式;

定义数据结构:基于数据行类型模型,定义数据源的数据行的字段结构信息;

管理分区:对接入的数据源数据进行分区,从而实现并行处理;

统一不同数据源API:不同数据源产生的记录都使用统一的数据行进行表示,对记录的处理提供统一的API。

8.根据权利要求5所述的解耦计算引擎的异构数据源的接入方法,其特征在于,所述数据源动态分区定义接口:用于动态地产出数据源的分区,统一管理不同数据源的分区规则,为上层应用提供一致的分区接口。

9.根据权利要求5所述的解耦计算引擎的异构数据源的接入方法,其特征在于,所述数据源动态分区定义接口的功能包括:

定义分区枚举接口:提供next()方法按顺序返回分区对象;

提供分区描述:分区描述包含分区标识、路径范围描述分区概要信息;

支持动态分区:分区集可以不固定,支持动态获取更多新分区;

并发安全:多线程下分区访问保证线程安全;

数据源透明:由分区枚举接口隐藏。

10.根据权利要求5所述的解耦计算引擎的异构数据源的接入方法,其特征在于,所述分区数据读取接口,封装不同数据源的读取细节差异,为上层应用提供统一高性能的数据读取服务;

所述分区数据读取接口用于真正读取和处理源中每一个分区对象的数据记录,每个分区读取输出数据的列表集合,列表的数据类型为统一数据行格式;

所述分区数据读取接口的实现是针对具体的数据源来制定的,不同源有不同的分区数据读取实现。

11.根据权利要求5所述的解耦计算引擎的异构数据源的接入方法,其特征在于,所述分区数据读取接口的功能包括:

提供数据读取接口:从数据源并行读出记录,封装为统一数据行格式返回;

遮掩数据源差异:提供统一的读取API接口;

并发控制:支持多线程并发读取保证读取性能;

容错机制:处理源异常能重试或者失败转异常输出。

12.根据权利要求5所述的解耦计算引擎的异构数据源的接入方法,其特征在于,所述数据源存储定义接口,描述数据源存储的元数据和基本行为,规范如何连接和操作各种数据源的存储,为定制各类数据源的存储提供通用标准接口规范。

13.根据权利要求5所述的解耦计算引擎的异构数据源的接入方法,其特征在于,所述数据源存储定义接口的具体流程包括:

设置接收数据的数据行的字段结构定义,包括:字段名称、字段的数据类型;

传递数据分区对象对应的子任务的索引号到分区数据写入接口,以支持并行写入多个分区;

创建分区数据写入接口。

14.根据权利要求5所述的解耦计算引擎的异构数据源的接入方法,其特征在于,所述分区数据写入接口,封装写入不同数据源的细节差异,为上层应用提供统一高性能的数据写入服务;所述分区数据写入接口用于真正写入中每一个分区对象的数据记录,每个分区数据的输入是分区对象的数据列表集合,列表的数据类型为统一数据行格式;所述分区数据写入接口是针对具体的数据源来制定的,不同源有不同的分区数据写入实现。

15.根据权利要求5所述的解耦计算引擎的异构数据源的接入方法,其特征在于,所述分区数据写入接口的功能包括:

提供数据写入接口:接收分区对象的数据,接收的数据为统一数据行格式,根据不同的数据源的类型,执行数据类型的转换,写入目标存储;

遮掩数据源差异:无论数据源是何种形式,提供统一的写入API接口;

并发控制:支持多线程并发写入,保证写入性能;

容错机制:数据写入失败,重试或者失败转异常输出。

16.根据权利要求1所述的解耦计算引擎的异构数据源的接入方法,其特征在于,所述S03包括:

S031、在数据读取时,将通过数据源读取标准接口接入的数据转换为计算引擎对应的格式,接入的数据具有多个数据分区对象,从而可以实现分布式的数据接入;

S032、在数据写入时,将计算引擎处理后的数据格式转换为统一的数据行格式,通过数据存储标准接口接收数据,写入各类数据存储。

17.一种解耦计算引擎的异构数据源的接入装置,其特征在于,该装置包括:

数据抽象模块,抽象统一的数据行,在数据行类型模型、数据行模型和数据类型方面进行抽象,从而实现各类异构数据源的统一接入和存储,在结构、功能方面具备扩展能力;

数据读写模块,数据的读取和写入基于插件机制,各类数据源分别实现各自的数据源读取和数据写入的插件,插件的实现规范则通过Java的父级类进行定义;

引擎适配模块,计算引擎的适配,计算引擎通过引擎各自的数据读取和写入的定义接

口,定义一种专用的数据源。

18.一种计算机设备,包括存储器、处理器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实现权利要求1-16任一项所述方法。

19.一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存储有执行权利要求1-16任一项所述方法的计算机程序。

# 一种解耦计算引擎的异构数据源的接入方法及装置

## 技术领域

[0001] 本发明涉及数据源领域,尤其是一种解耦计算引擎的异构数据源的接入方法及装置。

## 背景技术

[0002] 在生产系统中,存在大量异构的数据源,如MySQL表、Kafka主题、Elasticsearch索引、图数据库的节点和边、Hive表、对象存储文件等。同时,企业通常会采用多个计算引擎并行工作,如Spark、Flink、Presto等。

[0003] 当前的情况存在一些痛点问题:

[0004] 代码冗余和维护困难:为了接入不同的数据源,需要针对每个计算引擎分别开发数据源的插件。例如,如果要处理图数据库的点/边数据,就需要开发Spark、Flink和Presto的数据读取和写入插件,导致存在三份代码。这种重复开发和维护不仅增加了工作量,还增加了出错的可能性。

[0005] 参数不一致:由于每个计算引擎都有自己的数据源插件,每个插件可能会有不同的参数和配置方式。这导致在接入数据源时,需要针对不同的计算引擎使用不同的参数设置,给开发人员带来了困扰,并增加了使用和维护的复杂性。

[0006] 使用成本高:由于需要为每个计算引擎开发和维护数据源插件,企业需要投入大量的时间和资源。此外,由于插件代码的冗余和维护难度,对于新的数据源接入或计算引擎的更换,也需要进行重复的开发和调试工作。

## 发明内容

[0007] 为解决现有技术存在的问题,本发明提供一种解耦计算引擎的异构数据源的接入方法及装置,使用统一的API来处理数据源的查询和写入操作,从而解耦不同计算引擎间的依赖关系,支持多种计算引擎,大大降低开发和维护的成本。

[0008] 为实现上述目的,本发明采用下述技术方案:

[0009] 在本发明一实施例中,提出了一种解耦计算引擎的异构数据源的接入方法,该方法包括:

[0010] S01、抽象统一的数据行,在数据行类型模型、数据行模型和数据类型方面进行抽象,从而实现各类异构数据源的统一接入和存储,在结构、功能方面具备扩展能力;

[0011] 进一步地,所述S01包括:

[0012] S011、抽象数据行类型模型描述数据行中的字段、各字段的属性,字段的属性包括:字段的名称、数据类型、是否允许为空值等,相当于表结构的定义;

[0013] S012、数据行模型定义行数据的信息,在数据源的读取和写入过程中,用于交换和传输数据;

[0014] 进一步地,所述数据行模型包括:

[0015] 行数据的类型定义,通过数据行类型模型定义行数据对应的字段信息;

[0016] 对象数组,存储记录的各个字段值,每个位置对应一个字段;

[0017] 字段访问索引,通过索引直接访问对象数组内容,而不是通过Java反射访问数组,性能更高;

[0018] 字段名称映射,保证行类型模型定义中的字段名与对象数组位置对应,支持灵活访问;

[0019] 序列化器,由数据行类型模型与数据行建立关联并负责数据行的序列化与反序列化工作,从而实现数据高效传输;

[0020] 元数据,记录数据来源、长度、版本等辅助描述数据行的信息;

[0021] 持久化处理器,支持数据行持久化为文件或数据库表等,通过转换数据行类型实现。

[0022] S013、抽象字段的统一数据类型,字段的数据类型在数据行类型中表示一个字段的数据类型;建立各个数据源的数据类型和统一数据类型之间的映射关系,从而可以使用统一的API实现数据的接入和存储;映射关系的建立是由各个数据源读取和写入插件完成的;在数据源读取时,将数据源的数据类型转换为统一数据类型。

[0023] 进一步地,所述数据类型包括:

[0024] 基本数据类型包括:整型、字符串、布尔等;

[0025] 复合数据类型:多个基本数据类型组成的数据类型;例如,由整型和字符串两种基本数据类型组成的复合数据类型。

[0026] 数组类型:定义数组的数据类型,数据的元素的数据类型通过基本的数据类型进行定义;

[0027] Map映射数据类型:用于定义由key和value之间关系的数据类型,分别定义里key和value的数据类型。

[0028] S02、数据的读取和写入基于插件机制,插件机制的实现有多种方式;MySQL、Elasticsearch等各类数据源分别实现各自的数据源读取和数据写入的插件,插件的实现规范则通过Java的父级类进行定义。

[0029] 进一步地,所述S02包括:

[0030] S021、数据源读取标准接口,利用数据源读取定义接口,限定数据源的元数据和基本行为,数据源读取标准接口包括:数据源读取定义接口、数据源动态分区定义接口与分区数据读取接口;

[0031] 数据源读取定义接口负责描述整个数据源,并创建数据源动态分区定义接口和分区数据读取接口;

[0032] 数据源动态分区定义接口负责不断产出分区对象描述每个分区,分区对象由调度框架分配给不同任务执行子任务;

[0033] 分区数据读取接口由子任务调用,读取分配给子任务的分区对象的数据,进行真实的数据记录的读取和处理。

[0034] MySQL、Elasticsearch等各类数据源分别实现如上三类接口,形成了一个完整的数据源到具备并行处理的统一数据行格式数据集,提高了数据源数据读取的抽象能力和通用性。

[0035] 进一步地,所述数据源读取定义接口:利用数据源读取定义接口,描述数据源的元

数据和基本行为。

[0036] 进一步地,所述数据源读取定义接口的功能包括:

[0037] 定义数据源的连接参数:提供连接地址、配置等信息配置数据源如Kafka、Hive地址等;

[0038] 提供数据读取接口:从数据源读取数据,转换为统一的数据行格式;

[0039] 定义数据结构:基于数据行类型模型,定义数据源的数据行的字段结构信息;

[0040] 管理分区:对接入的数据源数据进行分区,从而实现并行处理;

[0041] 统一不同数据源API:不同数据源产生的记录都使用统一的数据行进行表示,对记录的处理提供统一的API,如map、filter等,而不关心数据来自何处。

[0042] 进一步地,所述数据源动态分区定义接口:用于动态地产出数据源的分区,统一管理不同数据源(如Kafka、HDFS等)的分区规则,为上层应用提供一致的分区接口;

[0043] 动态地产出数据源的分区相当于对需要读取的数据源进行切分,划分为多个数据源的分区;每个数据源的分区对应一个并发任务,从而可以实现分布式地并行处理;

[0044] 进一步地,所述数据源动态分区定义接口的功能包括:

[0045] 定义分区枚举接口:提供next()方法按顺序返回分区对象;

[0046] 提供分区描述:分区描述包含分区标识、路径范围等描述分区概要信息;

[0047] 支持动态分区:分区集可以不固定,需要支持动态获取更多新分区;

[0048] 并发安全:多线程下分区访问需保证线程安全;

[0049] 数据源透明:对接数据源无需关心其内部分区规则,由分区枚举接口隐藏。

[0050] 进一步地,所述分区数据读取接口,封装不同数据源的读取细节差异,为上层应用提供统一高性能的数据读取服务;分区数据读取接口用于真正读取和处理源中每一个分区对象的数据记录,每个分区读取输出数据的列表集合,列表的数据类型为统一数据行格式;分区数据读取接口的实现是针对具体的数据源来制定的,不同源有不同的分区数据读取实现;

[0051] 进一步地,所述分区数据读取接口的功能包括:

[0052] 提供数据读取接口:从数据源并行读出记录,封装为统一数据行格式返回;

[0053] 遮掩数据源差异:无论数据源是何种形式,提供统一的读取API接口;

[0054] 并发控制:支持多线程并发读取保证读取性能;

[0055] 容错机制:处理源异常能重试或者失败转异常输出。

[0056] S022、数据存储标准接口,接收数据并写入各类数据存储,数据存储标准接口包括:数据源存储定义接口、分区数据写入接口;

[0057] 数据源存储定义接口负责描述整个数据源存储,创建分区数据写入接口;

[0058] 分区数据写入接口接收分区对象的数据,每个分区对象由调度框架分配一个任务执行的子任务,子任务调用分区数据写入接口,将分区对象的数据写入目标存储。

[0059] 进一步地,所述数据源存储定义接口,描述数据源存储的元数据和基本行为,规范如何连接和操作各种数据源的存储,为定制各类数据源的存储提供通用标准接口规范:

[0060] 进一步地,所述数据源存储定义接口的具体流程包括:

[0061] 设置接收数据的数据行的字段结构定义,包括:字段名称、字段的数据类型等;

[0062] 传递数据分区对象对应的子任务的索引号到分区数据写入接口,以支持并行写入

多个分区。

[0063] 创建分区数据写入接口。

[0064] 进一步地,所述分区数据写入接口,封装写入不同数据源的细节差异,为上层应用提供统一高性能的数据写入服务;所述分区数据写入接口用于真正写入中每一个分区对象的数据记录,每个分区数据的输入是分区对象的数据列表集合,列表的数据类型为统一数据行格式;所述分区数据写入接口是针对具体的数据源来制定的,不同源有不同的分区数据写入实现;

[0065] 进一步地,所述分区数据写入接口的功能包括:

[0066] 提供数据写入接口:接收分区对象的数据,接收的数据为统一数据行格式,根据不同的数据源的类型,执行数据类型的转换,写入目标存储;

[0067] 遮掩数据源差异:无论数据源是何种形式,提供统一的写入API接口;

[0068] 并发控制:支持多线程并发写入,保证写入性能;

[0069] 容错机制:数据写入失败,重试或者失败转异常输出等。

[0070] S03、计算引擎的适配,Spark、Flink、Presto等计算引擎通过引擎各自的数据读取和写入的定义接口,定义一种专用的数据源。

[0071] 进一步地,所述S03包括:

[0072] S031、在数据读取时,将通过数据源读取标准接口接入的数据转换为计算引擎对应的格式,接入的数据具有多个数据分区对象,从而可以实现分布式的数据接入;

[0073] S032、在数据写入时,将计算引擎处理后的数据格式转换为统一的数据行格式,通过数据存储标准接口接收数据,写入各类数据存储。

[0074] 在本发明一实施例中,还提出了一种解耦计算引擎的异构数据源的接入装置,该装置包括:

[0075] 数据抽象模块,抽象统一的数据行,在数据行类型模型、数据行模型和数据类型方面进行抽象,从而实现各类异构数据源的统一接入和存储,在结构、功能方面具备扩展能力;

[0076] 数据读写模块,数据的读取和写入基于插件机制,各类数据源分别实现各自的数据源读取和数据写入的插件,插件的实现规范则通过Java的父级类进行定义;

[0077] 引擎适配模块,计算引擎的适配,计算引擎通过引擎各自的数据读取和写入的定义接口,定义一种专用的数据源。

[0078] 在本发明一实施例中,还提出了一种计算机设备,包括存储器、处理器及存储在存储器上并可在处理器上运行的计算机程序,处理器执行计算机程序时实现前述解耦计算引擎的异构数据源的接入方法。

[0079] 在本发明一实施例中,还提出了一种计算机可读存储介质,计算机可读存储介质存储有执行解耦计算引擎的异构数据源的接入方法的计算机程序。

[0080] 有益效果:

[0081] 本发明一种解耦计算引擎的异构数据源的接入方法及装置,支持多种计算引擎,每种数据源的读取和写入只需开发一套代码;通过引入统一规范的接口,每种数据源的读取和写入只需针对这套标准接口进行一次开发;遵循标准的计算引擎可以直接使用此连接器,大大降低开发和维护的成本;支持高性能的分布式数据处理,标准接口定义了分区读取

和写入等能力,按分区方式同时处理多个分区内的数据,充分利用计算资源并行性,从而支持高吞吐量的大规模数据处理任务。

## 附图说明

[0082] 图1是本发明解耦计算引擎的异构数据源的接入方法流程示意图;

[0083] 图2是本发明解耦计算引擎的异构数据源的接入装置结构示意图;

[0084] 图3是本发明一实施例的计算机设备结构示意图。

## 具体实施方式

[0085] 下面将参考若干示例性实施方式来描述本发明的原理和精神,应当理解,给出这些实施方式仅仅是为了使本领域技术人员能够更好地理解进而实现本发明,而并非以任何方式限制本发明的范围。相反,提供这些实施方式是为了使本公开更加透彻和完整,并且能够将本公开的范围完整地传达给本领域的技术人员。

[0086] 本领域技术人员知道,本发明的实施方式可以实现为一种系统、装置、设备、方法或计算机程序产品。因此,本公开可以具体实现为以下形式,即:完全的硬件、完全的软件(包括固件、驻留软件、微代码等),或者硬件和软件结合的形式。

[0087] 本发明涉及的专业名词及其解释:

[0088] 根据本发明的实施方式,提出了一种解耦计算引擎的异构数据源的接入方法及装置,使用统一的API来处理数据源的查询和写入操作,从而解耦不同计算引擎间的依赖关系,支持多种计算引擎,大大降低开发和维护的成本。

[0089] 下面参考本发明的若干代表性实施方式,详细阐释本发明的原理和精神。

[0090] 如图1所示,本发明涉及的一种解耦计算引擎的异构数据源的接入方法,该方法包括:

[0091] S01、抽象统一的数据行,在数据行类型模型、数据行模型和数据类型方面进行抽象,从而实现各类异构数据源的统一接入和存储,在结构、功能方面具备扩展能力;

[0092] 进一步地,所述S01包括:

[0093] S011、抽象数据行类型模型描述数据行中的字段、各字段的属性,字段的属性包括:字段的名称、数据类型、是否允许为空值等,相当于表结构的定义;

[0094] S012、数据行模型定义行数据的信息,在数据源的读取和写入过程中,用于交换和传输数据;

[0095] 进一步地,所述数据行模型包括:

[0096] 行数据的类型定义,通过数据行类型模型定义行数据对应的字段信息;

[0097] 对象数组,存储记录的各个字段值,每个位置对应一个字段;

[0098] 字段访问索引,通过索引直接访问对象数组内容,而不是通过Java反射访问数组,性能更高;

[0099] 字段名称映射,保证行类型模型定义中的字段名与对象数组位置对应,支持灵活访问;

[0100] 序列化器,由数据行类型模型与数据行建立关联并负责数据行的序列化与反序列化工作,从而实现数据高效传输;

[0101] 元数据,记录数据来源、长度、版本等辅助描述数据行的信息;

[0102] 持久化处理器,支持数据行持久化为文件或数据库表等,通过转换数据行类型实现。

[0103] S013、抽象字段的统一数据类型,字段的数据类型在数据行类型中表示一个字段的数据类型;建立各个数据源的数据类型和统一数据类型之间的映射关系,从而可以使用统一的API实现数据的接入和存储;映射关系的建立是由各个数据源读取和写入插件完成的;在数据源读取时,将数据源的数据类型转换为统一数据类型。

[0104] 进一步地,所述数据类型包括:

[0105] 基本数据类型包括:整型、字符串、布尔等;

[0106] 复合数据类型,多个基本数据类型组成的数据类型;例如,由整型和字符串两种基本数据类型组成的复合数据类型。

[0107] 数组类型:定义数组的数据类型,数据的元素的数据类型通过基本的数据类型进行定义;

[0108] Map映射数据类型,用于定义由key和value之间关系的数据类型,分别定义里key和value的数据类型。

[0109] S02、数据的读取和写入基于插件机制,插件机制的实现有多种方式,例如,使用工厂设计模式、模板设计模式等均可实现插件的扩展。MySQL、Elasticsearch等各类数据源分别实现各自的数据源读取和数据写入的插件,插件的实现规范则通过Java的父级类进行定义。

[0110] 进一步地,所述S02包括:

[0111] S021、数据源读取标准接口,利用数据源读取定义接口,限定数据源的元数据和基本行为,数据源读取标准接口包括:数据源读取定义接口、数据源动态分区定义接口与分区数据读取接口;

[0112] 数据源读取定义接口负责描述整个数据源,并创建数据源动态分区定义接口和分区数据读取接口;

[0113] 数据源动态分区定义接口负责不断产出分区对象描述每个分区,分区对象由调度框架分配给不同任务执行子任务;

[0114] 分区数据读取接口由子任务调用,读取分配给子任务的分区对象的数据,进行真实的数据记录的读取和处理。

[0115] MySQL、Elasticsearch等各类数据源分别实现如上三类接口,形成了一个完整的数据源到具备并行处理的统一数据行格式数据集,提高了数据源数据读取的抽象能力和通用性。

[0116] 进一步地,所述数据源读取定义接口:利用数据源读取定义接口,描述数据源的元数据和基本行为。

[0117] 进一步地,所述数据源读取定义接口的功能包括:

[0118] 定义数据源的连接参数:提供连接地址、配置等信息配置数据源如Kafka、Hive地址等;

[0119] 提供数据读取接口:从数据源读取数据,转换为统一的数据行格式;

[0120] 定义数据结构:基于数据行类型模型,定义数据源的数据行的字段结构信息;

[0121] 管理分区:对接入的数据源数据进行分区,从而实现并行处理;

[0122] 统一不同数据源API:不同数据源产生的记录都使用统一的数据行进行表示,对记录的处理提供统一的API,如map、filter等,而不关心数据来自何处。

[0123] 进一步地,所述数据源动态分区定义接口:用于动态地产出数据源的分区,统一管理不同数据源(如Kafka、HDFS等)的分区规则,为上层应用提供一致的分区接口;

[0124] 动态地产出数据源的分区相当于对需要读取的数据源进行切分,划分为多个数据源的分区;每个数据源的分区对应一个并发任务,从而可以实现分布式地并行处理;

[0125] 进一步地,所述数据源动态分区定义接口的功能包括:

[0126] 定义分区枚举接口:提供next()方法按顺序返回分区对象;

[0127] 提供分区描述:分区描述包含分区标识、路径范围等描述分区概要信息;

[0128] 支持动态分区:分区集可以不固定,需要支持动态获取更多新分区;

[0129] 并发安全:多线程下分区访问需保证线程安全;

[0130] 数据源透明:对接数据源无需关心其内部分区规则,由分区枚举接口隐藏。

[0131] 进一步地,所述分区数据读取接口,封装不同数据源的读取细节差异,为上层应用提供统一高性能的数据读取服务;分区数据读取接口用于真正读取和处理源中每一个分区对象的数据记录,每个分区读取输出数据的列表集合,列表的数据类型为统一数据行格式;分区数据读取接口的实现是针对具体的数据源来制定的,不同源有不同的分区数据读取实现;

[0132] 进一步地,所述分区数据读取接口的功能包括:

[0133] 提供数据读取接口:从数据源并行读出记录,封装为统一数据行格式返回;

[0134] 遮掩数据源差异:无论数据源是何种形式,提供统一的读取API接口;

[0135] 并发控制:支持多线程并发读取保证读取性能;

[0136] 容错机制:处理源异常能重试或者失败转异常输出。

[0137] S022、数据存储标准接口,接收数据并写入各类数据存储,数据存储标准接口包括:数据源存储定义接口、分区数据写入接口;

[0138] 数据源存储定义接口负责描述整个数据源存储,创建分区数据写入接口;

[0139] 分区数据写入接口接收分区对象的数据,每个分区对象由调度框架分配一个任务执行的子任务,子任务调用分区数据写入接口,将分区对象的数据写入目标存储。

[0140] 进一步地,所述数据源存储定义接口,描述数据源存储的元数据和基本行为,规范如何连接和操作各种数据源的存储,为定制各类数据源的存储提供通用标准接口规范:

[0141] 进一步地,所述数据源存储定义接口的具体流程包括:

[0142] 设置接收数据的数据行的字段结构定义,包括:字段名称、字段的数据类型等;

[0143] 传递数据分区对象对应的子任务的索引号到分区数据写入接口,以支持并行写入多个分区。

[0144] 创建分区数据写入接口。

[0145] 进一步地,所述分区数据写入接口,封装写入不同数据源的细节差异,为上层应用提供统一高性能的数据写入服务;所述分区数据写入接口用于真正写入中每一个分区对象的数据记录,每个分区数据的输入是分区对象的数据列表集合,列表的数据类型为统一数据行格式;所述分区数据写入接口是针对具体的数据源来制定的,不同源有不同的分区数

据写入实现;

[0146] 进一步地,所述分区数据写入接口的功能包括:

[0147] 提供数据写入接口:接收分区对象的数据,接收的数据为统一数据行格式,根据不同的数据源的类型,执行数据类型的转换,写入目标存储;

[0148] 遮掩数据源差异:无论数据源是何种形式,提供统一的写入API接口;

[0149] 并发控制:支持多线程并发写入,保证写入性能;

[0150] 容错机制:数据写入失败,重试或者失败转异常输出等。

[0151] S03、计算引擎的适配,Spark、Flink、Presto等计算引擎通过引擎各自的数据读取和写入的定义接口,定义一种专用的数据源。

[0152] 进一步地,所述S03包括:

[0153] S031、在数据读取时,将通过数据源读取标准接口接入的数据转换为计算引擎对应的格式,接入的数据具有多个数据分区对象,从而可以实现分布式的数据接入;

[0154] 例如,对于Spark计算引擎,统一的数据行转换为Spark的DataFrame数据集。对于Flink计算引擎,将统一的数据行转换为Flink的Dataset数据集。

[0155] S032、在数据写入时,将计算引擎处理后的数据格式转换为统一的数据行格式,通过数据存储标准接口接收数据,写入各类数据存储。

[0156] 例如,对于Spark计算引擎,处理后的数据格式为DataFrame数据集,将DataFrame格式的数据集转换为统一的数据行格式,再通过数据存储标准接口接收数据;

[0157] 对于Flink计算引擎,处理后的数据格式为Dataset数据集,将Dataset格式的数据集转换为统一的数据行格式,再通过数据存储标准接口接收数据

[0158] 由于Spark、Flink等天然具备分布式数据处理的能力,可以将Spark或者Flink的每个分区的数据映射为数据存储标准接口的分区对象的数据,从而可以实现分布式的数据写入。

[0159] 需要说明的是,尽管在上述实施例及附图中以特定顺序描述了本发明方法的操作,但是,这并非要求或者暗示必须按照该特定顺序来执行这些操作,或是必须执行全部所示的操作才能实现期望的结果。附加地或备选地,可以省略某些步骤,将多个步骤合并为一个步骤执行,和/或将一个步骤分解为多个步骤执行。

[0160] 为了对上述解耦计算引擎的异构数据源的接入方法进行更为清楚的解释,下面结合具体的实施例来进行说明,然而值得注意的是该实施例仅是为了更好地说明本发明,并不构成对本发明不当的限定。

[0161] S01、抽象统一的数据行,在数据行类型模型、数据行模型和数据类型方面进行抽象,从而实现各类异构数据源的统一接入和存储,在结构、功能方面具备扩展能力;

[0162] 所述S01包括:

[0163] S011、抽象数据行类型模型描述数据行中的字段、各字段的属性,字段的属性包括:字段的名称、数据类型、是否允许为空值等,相当于表结构的定义;

[0164] S012、数据行模型定义行数据的信息,在数据源的读取和写入过程中,用于交换和传输数据;

[0165] 所述数据行模型包括:

[0166] 行数据的类型定义,通过数据行类型模型定义行数据对应的字段信息;

[0167] 对象数组,存储记录的各个字段值,每个位置对应一个字段;

[0168] 字段访问索引,通过索引直接访问对象数组内容,而不是通过Java反射访问数组,性能更高;

[0169] 字段名称映射,保证行类型模型定义中的字段名与对象数组位置对应,支持灵活访问;

[0170] 序列化器,由数据行类型模型与数据行建立关联并负责数据行的序列化与反序列化工作,从而实现数据高效传输;

[0171] 元数据,记录数据来源、长度、版本等辅助描述数据行的信息;

[0172] 持久化处理器,支持数据行持久化为文件或数据库表等,通过转换数据行类型实现。

[0173] S013、抽象字段的统一数据类型,字段的数据类型在数据行类型中表示一个字段的数据类型;建立各个数据源的数据类型和统一数据类型之间的映射关系,从而可以使用统一的API实现数据的接入和存储;映射关系的建立是由各个数据源读取和写入插件完成的;在数据源读取时,将数据源的数据类型转换为统一数据类型。

[0174] 所述数据类型包括:

[0175] 基本数据类型包括:整型、字符串、布尔等;

[0176] 复合数据类型,多个基本数据类型组成的数据类型;例如,由整型和字符串两种基本数据类型组成的复合数据类型。

[0177] 数组类型:定义数组的数据类型,数据的元素的数据类型通过基本的数据类型进行定义;

[0178] Map映射数据类型,用于定义由key和value之间关系的数据类型,分别定义里key和value的数据类型。

[0179] S02、数据的读取和写入基于插件机制,插件机制的实现有多种方式,例如,使用工厂设计模式、模板设计模式等均可实现插件的扩展。MySQL、Elasticsearch等各类数据源分别实现各自的数据源读取和数据写入的插件,插件的实现规范则通过Java的父级类进行定义。

[0180] 所述S02包括:

[0181] S021、数据源读取标准接口,利用数据源读取定义接口,限定数据源的元数据和基本行为,数据源读取标准接口包括:数据源读取定义接口、数据源动态分区定义接口与分区数据读取接口;

[0182] 数据源读取定义接口负责描述整个数据源,并创建数据源动态分区定义接口和分区数据读取接口;

[0183] 数据源动态分区定义接口负责不断产出分区对象描述每个分区,分区对象由调度框架分配给不同任务执行子任务;

[0184] 分区数据读取接口由子任务调用,读取分配给子任务的分区对象的数据,进行真实的数据记录的读取和处理。

[0185] MySQL、Elasticsearch等各类数据源分别实现如上三类接口,形成了一个完整的数据源到具备并行处理的统一数据行格式数据集,提高了数据源数据读取的抽象能力和通用性。

[0186] 所述数据源读取定义接口:利用数据源读取定义接口,描述数据源的元数据和基本行为。

[0187] 所述数据源读取定义接口的功能包括:

[0188] 定义数据源的连接参数:提供连接地址、配置等信息配置数据源如Kafka、Hive地址等;

[0189] 提供数据读取接口:从数据源读取数据,转换为统一的数据行格式;

[0190] 定义数据结构:基于数据行类型模型,定义数据源的数据行的字段结构信息;

[0191] 管理分区:对接入的数据源数据进行分区,从而实现并行处理;

[0192] 统一不同数据源API:不同数据源产生的记录都使用统一的数据行进行表示,对记录的处理提供统一的API,如map、filter等,而不关心数据来自何处。

[0193] 所述数据源动态分区定义接口:用于动态地产出数据源的分区,统一管理不同数据源(如Kafka、HDFS等)的分区规则,为上层应用提供一致的分区接口;

[0194] 动态地产出数据源的分区相当于对需要读取的数据源进行切分,划分为多个数据源的分区;每个数据源的分区对应一个并发任务,从而可以实现分布式地并行处理;

[0195] 所述数据源动态分区定义接口的功能包括:

[0196] 定义分区枚举接口:提供next()方法按顺序返回分区对象;

[0197] 提供分区描述:分区描述包含分区标识、路径范围等描述分区概要信息;

[0198] 支持动态分区:分区集可以不固定,需要支持动态获取更多新分区;

[0199] 并发安全:多线程下分区访问需保证线程安全;

[0200] 数据源透明:对接数据源无需关心其内部分区规则,由分区枚举接口隐藏。

[0201] 所述分区数据读取接口,封装不同数据源的读取细节差异,为上层应用提供统一高性能的数据读取服务;分区数据读取接口用于真正读取和处理源中每一个分区对象的数据记录,每个分区读取输出数据的列表集合,列表的数据类型为统一数据行格式;分区数据读取接口的实现是针对具体的数据源来制定的,不同源有不同的分区数据读取实现;

[0202] 所述分区数据读取接口的功能包括:

[0203] 提供数据读取接口:从数据源并行读出记录,封装为统一数据行格式返回;

[0204] 遮掩数据源差异:无论数据源是何种形式,提供统一的读取API接口;

[0205] 并发控制:支持多线程并发读取保证读取性能;

[0206] 容错机制:处理源异常能重试或者失败转异常输出。

[0207] S022、数据存储标准接口,接收数据并写入各类数据存储,数据存储标准接口包括:数据源存储定义接口、分区数据写入接口;

[0208] 数据源存储定义接口负责描述整个数据源存储,创建分区数据写入接口;

[0209] 分区数据写入接口接收分区对象的数据,每个分区对象由调度框架分配一个任务执行的子任务,子任务调用分区数据写入接口,将分区对象的数据写入目标存储。

[0210] 所述数据源存储定义接口,描述数据源存储的元数据和基本行为,规范如何连接和操作各种数据源的存储,为定制各类数据源的存储提供通用标准接口规范:

[0211] 所述数据源存储定义接口的具体流程包括:

[0212] 设置接收数据的数据行的字段结构定义,包括:字段名称、字段的数据类型等;

[0213] 传递数据分区对象对应的子任务的索引号到分区数据写入接口,以支持并行写入

多个分区。

[0214] 创建分区数据写入接口。

[0215] 所述分区数据写入接口,封装写入不同数据源的细节差异,为上层应用提供统一高性能的数据写入服务;所述分区数据写入接口用于真正写入中每一个分区对象的数据记录,每个分区数据的输入是分区对象的数据列表集合,列表的数据类型为统一数据行格式;所述分区数据写入接口是针对具体的数据源来制定的,不同源有不同的分区数据写入实现;

[0216] 所述分区数据写入接口的功能包括:

[0217] 提供数据写入接口:接收分区对象的数据,接收的数据为统一数据行格式,根据不同的数据源的类型,执行数据类型的转换,写入目标存储;

[0218] 遮掩数据源差异:无论数据源是何种形式,提供统一的写入API接口;

[0219] 并发控制:支持多线程并发写入,保证写入性能;

[0220] 容错机制:数据写入失败,重试或者失败转异常输出等。

[0221] S03、计算引擎的适配,Spark、Flink、Presto等计算引擎通过引擎各自的数据读取和写入的定义接口,定义一种专用的数据源。

[0222] 所述S03包括:

[0223] S031、在数据读取时,将通过数据源读取标准接口接入的数据转换为计算引擎对应的格式,接入的数据具有多个数据分区对象,从而可以实现分布式的数据接入;

[0224] 例如,对于Spark计算引擎,统一的数据行转换为Spark的DataFrame数据集。对于Flink计算引擎,将统一的数据行转换为Flink的Dataset数据集。

[0225] S032、在数据写入时,将计算引擎处理后的数据格式转换为统一的数据行格式,通过数据存储标准接口接收数据,写入各类数据存储。

[0226] 例如,对于Spark计算引擎,处理后的数据格式为DataFrame数据集,将DataFrame格式的数据集转换为统一的数据行格式,再通过数据存储标准接口接收数据;

[0227] 对于Flink计算引擎,处理后的数据格式为Dataset数据集,将Dataset格式的数据集转换为统一的数据行格式,再通过数据存储标准接口接收数据

[0228] 由于Spark、Flink等天然具备分布式数据处理的能力,可以将Spark或者Flink的每个分区的数据映射为数据存储标准接口的分区对象的数据,从而可以实现分布式的数据写入。

[0229] 以Spark计算引擎为例,读取数据的代码如下:

```
// 1. 定义 MySQL 数据库的主机、端口、数据库、表等信息
val optionMap = Map(
  "host" -> "192.168.118.11",     // 指定数据库的主机
  "port" -> "11",                 // 指定数据库的端口
  "database" -> "default",        // 指定数据库名称
  "table" -> "access_log"         // 指定需要读取的表
)
val data = spark.read.format("seacrab")  // 通过 format 指定数据源的名称，专门用于读取"数据源读取标准接口"定义的数据源
  .options(optionMap)             // 通过 optionMap，引入上面定义 MySQL 数据库的主机、端口等信息
  .load()                         // 加载动作，加载数据
```

[0230]

[0231] 通过format方法指定数据源的名称,专门用于读取"数据源读取标准接口"定义的数据源,并通过options方法指定MySQL数据库的主机、端口、数据库名称、表等信息,通过load方法加载数据,从而实现加载数据库的表,而不用实现复杂的数据读取逻辑。

[0232] 以Spark计算引擎为例,数据写入的代码如下:

```
// 1. 定义MySQL数据库的主机、端口、数据库、表等信息
val optionMap = Map(
  "host" -> "192.168.118.11",     // 指定数据库的主机
  "port" -> "11",                 // 指定数据库的端口
  "database" -> "default",        // 指定数据库名称
  "table" -> "access_log"         // 指定需要读取的表
  "save_mode" -> "overwrite"      // 指定数据写入的模式：
例如，覆盖方式使用overwrite，追加方式使用append。
)
// 将data数据写入数据库
data.write.format("seacrab")  // 通过format指定数据源
的名称，专门用于将数据转换统一的数据格式，由"数据存储标准接
口"接收数据并执行数据的写入
.options(optionMap)           // 通过optionMap，引入上
面定义MySQL数据库的主机、端口等信息
.save()                       // 执行数据写入的操作
```

[0233]

[0234] 将data数据写入数据库,通过format指定数据源的名称,专门用于将数据转换统一的数据格式,由数据存储标准接口接收数据并执行数据的写入,并通过options方法指定MySQL数据库的主机、端口、数据库名称、表、数据写入方式等信息,通过save方法执行数据写入的操作,从而实现将数据写入数据库的表,而不用实现复杂的数据写入逻辑。

[0235] 基于同一发明构思,本发明还提出一种解耦计算引擎的异构数据源的接入装置。该装置的实施可以参见上述方法的实施,重复之处不再赘述。以下所使用的术语“模块”,可以是实现预定功能的软件和/或硬件的组合。尽管以下实施例所描述的装置较佳地以软件来实现,但是硬件,或者软件和硬件的组合的实现也是可能并被构想的。

[0236] 图2是本发明解耦计算引擎的异构数据源的接入装置结构示意图。如图2所示,该装置包括:

[0237] 数据抽象模块110,抽象统一的数据行,在数据行类型模型、数据行模型和数据类

型方面进行抽象,从而实现各类异构数据源的统一接入和存储,在结构、功能方面具备扩展能力;

[0238] 数据读写模块120,数据的读取和写入基于插件机制,各类数据源分别实现各自的数据源读取和数据写入的插件,插件的实现规范则通过Java的父级类进行定义;

[0239] 引擎适配模块130,计算引擎的适配,计算引擎通过引擎各自的数据读取和写入的定义接口,定义一种专用的数据源。

[0240] 应当注意,尽管在上文详细描述中提及了解耦计算引擎的异构数据源的接入装置的若干模块,但是这种划分仅仅是示例性的并非强制性的。实际上,根据本发明的实施方式,上文描述的两个或更多模块的特征和功能可以在一个模块中具体化。反之,上文描述的一个模块的特征和功能可以进一步划分为由多个模块来具体化。

[0241] 基于前述发明构思,如图3所示,本发明还提出一种计算机设备200,包括存储器210、处理器220及存储在存储器210上并可在处理器220上运行的计算机程序230,处理器220执行计算机程序230时实现前述解耦计算引擎的异构数据源的接入方法。

[0242] 基于前述发明构思,本发明还提出一种计算机可读存储介质,计算机可读存储介质存储有执行前述解耦计算引擎的异构数据源的接入方法的计算机程序。

[0243] 本发明一种解耦计算引擎的异构数据源的接入方法及装置,支持多种计算引擎,每种数据源的读取和写入只需开发一套代码;通过引入统一规范的接口,每种数据源的读取和写入只需针对这套标准接口进行一次开发;遵循标准的计算引擎可以直接使用此连接器,大大降低开发和维护的成本;支持高性能的分布式数据处理,标准接口定义了分区读取和写入等能力,按分区方式同时处理多个分区内的数据,充分利用计算资源并行性,从而支持高吞吐量的大规模数据处理任务。

[0244] 虽然已经参考若干具体实施方式描述了本发明的精神和原理,但是应该理解,本发明并不限于所公开的具体实施方式,对各方面的划分也不意味着这些方面中的特征不能组合以进行受益,这种划分仅是为了表述的方便。本发明旨在涵盖所附权利要求的精神和范围内所包含的各种修改和等同布置。

[0245] 本文中以上描述的系统和技术的各种实施方式可以在数字电子电路系统、集成电路系统、现场可编程门阵列(FPGA)、专用集成电路(ASIC)、专用标准产品(ASSP)、芯片上系统的系统(SOC)、复杂可编程逻辑设备(CPLD)、计算机硬件、固件、软件、和/或它们的组合中实现。这些各种实施方式可以包括:实施在一个或者多个计算机程序中,该一个或者多个计算机程序可在包括至少一个可编程处理器的可编程系统上执行和/或解释,该可编程处理器可以是专用或者通用可编程处理器,可以从存储系统、至少一个输入装置、和至少一个输出装置接收数据和指令,并且将数据和指令传输至该存储系统、该至少一个输入装置、和该至少一个输出装置。

[0246] 用于实施本公开的方法的程序代码可以采用一个或多个编程语言的任何组合来编写。这些程序代码可以提供给通用计算机、专用计算机或其他可编程数据处理装置的处理器或控制器,使得程序代码当由处理器或控制器执行时使流程图和/或框图中所规定的功能/操作被实施。程序代码可以完全在机器上执行、部分地在机器上执行,作为独立软件包部分地在机器上执行且部分地在远程机器上执行或完全在远程机器或服务器上执行。

[0247] 在本公开的上下文中,机器可读介质可以是有形的介质,其可以包含或存储以供

指令执行系统、装置或设备使用或与指令执行系统、装置或设备结合地使用的程序。机器可读介质可以是机器可读信号介质或机器可读储存介质。机器可读介质可以包括但不限于电子的、磁性的、光学的、电磁的、红外的、或半导体系统、装置或设备，或者上述内容的任何合适组合。机器可读存储介质的更具体示例会包括基于一个或多个线的电气连接、便携式计算机盘、硬盘、随机存取存储器(RAM)、只读存储器(ROM)、可擦除可编程只读存储器(EPROM或快闪存储器)、光纤、便捷式紧凑盘只读存储器(CD-ROM)、光学储存设备、磁储存设备、或上述内容的任何合适组合。

[0248] 为了提供与用户的交互，可以在计算机上实施此处描述的系统和技术，该计算机具有：用于向用户显示信息的显示装置(例如，CRT(阴极射线管)或者LCD(液晶显示器)监视器)；以及键盘和指向装置(例如，鼠标或者轨迹球)，用户可以通过该键盘和该指向装置来将输入提供给计算机。其它种类的装置还可以用于提供与用户的交互；例如，提供给用户的反馈可以是任何形式的传感反馈(例如，视觉反馈、听觉反馈、或者触觉反馈)；并且可以用任何形式(包括声输入、语音输入或者、触觉输入)来接收来自用户的输入。

[0249] 可以将此处描述的系统和技术实施在包括后台部件的计算系统(例如，作为数据服务器)、或者包括中间件部件的计算系统(例如，应用服务器)、或者包括前端部件的计算系统(例如，具有图形用户界面或者网络浏览器的用户计算机，用户可以通过该图形用户界面或者该网络浏览器来与此处描述的系统和技术的实施方式交互)、或者包括这种后台部件、中间件部件、或者前端部件的任何组合的计算系统中。可以通过任何形式或者介质的数字数据通信(例如，通信网络)来将系统的部件相互连接。通信网络的示例包括：局域网(LAN)、广域网(WAN)和互联网。

[0250] 计算机系统可以包括客户端和服务器。客户端和服务器一般远离彼此并且通常通过通信网络进行交互。通过在相应的计算机上运行并且彼此具有客户端-服务器关系的计算机程序来产生客户端和服务器的关系。服务器可以是云服务器，也可以为分布式系统的服务器，或者是结合了区块链的服务器。

[0251] 应该理解，可以使用上面所示的各种形式的流程，重新排序、增加或删除步骤。例如，本发公开中记载的各步骤可以并行地执行也可以顺序地执行也可以不同的次序执行，只要能够实现本公开公开的技术方案所期望的结果，本文在此不进行限制。

[0252] 上述具体实施方式，并不构成对本公开保护范围的限制。本领域技术人员应该明白的是，根据设计要求和其他因素，可以进行各种修改、组合、子组合和替代。任何在本公开的精神和原则之内所作的修改、等同替换和改进等，均应包含在本公开保护范围之内。

[0253] 对本发明保护范围的限制，所属领域技术人员应该明白，在本发明的技术方案的基础上，本领域技术人员不需要付出创造性劳动即可做出的各种修改或变形仍在本发明的保护范围以内。

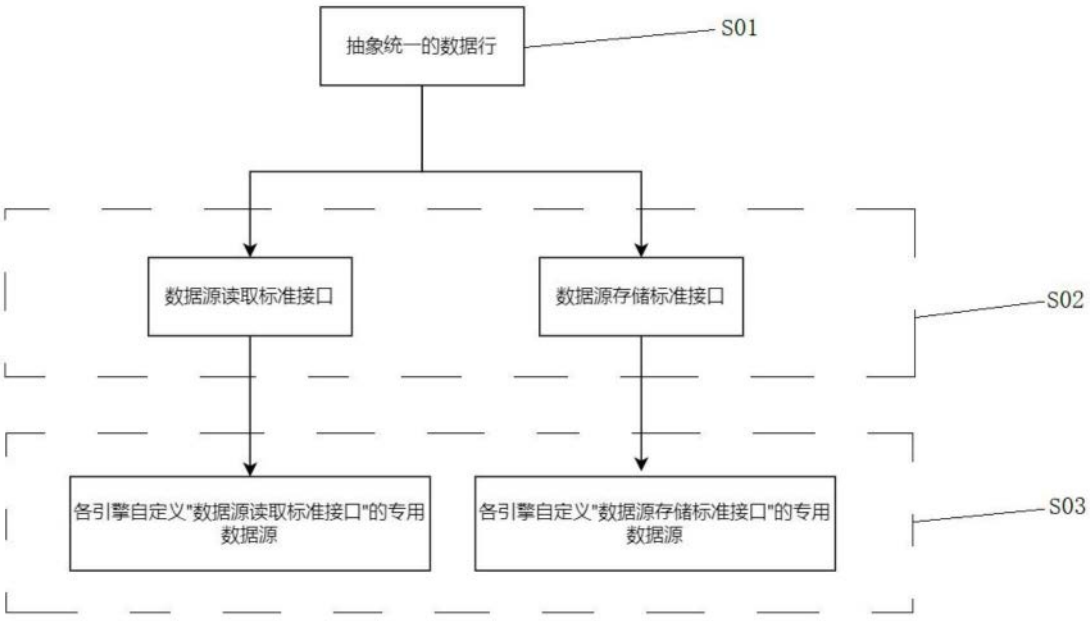

抽象统一的数据行
S01
数据源读取标准接口
数据源存储标准接口
S02
各引擎自定义"数据源读取标准接口"的专用数据源
各引擎自定义"数据源存储标准接口"的专用数据源
S03

图1

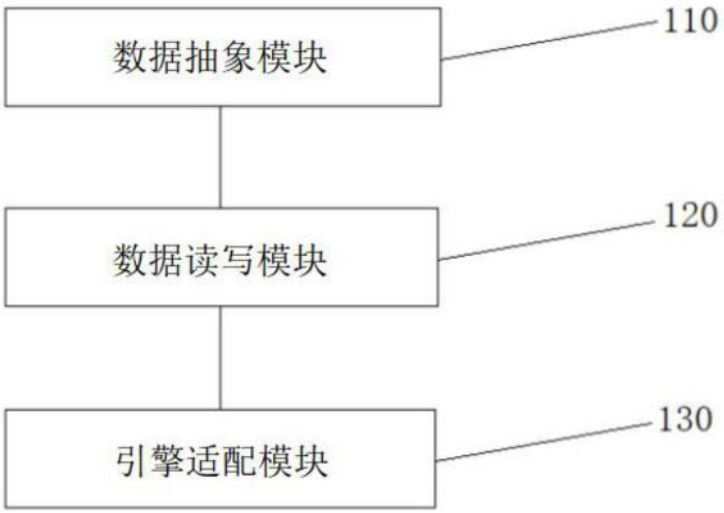

数据抽象模块
110
数据读写模块
120
引擎适配模块
130

图2

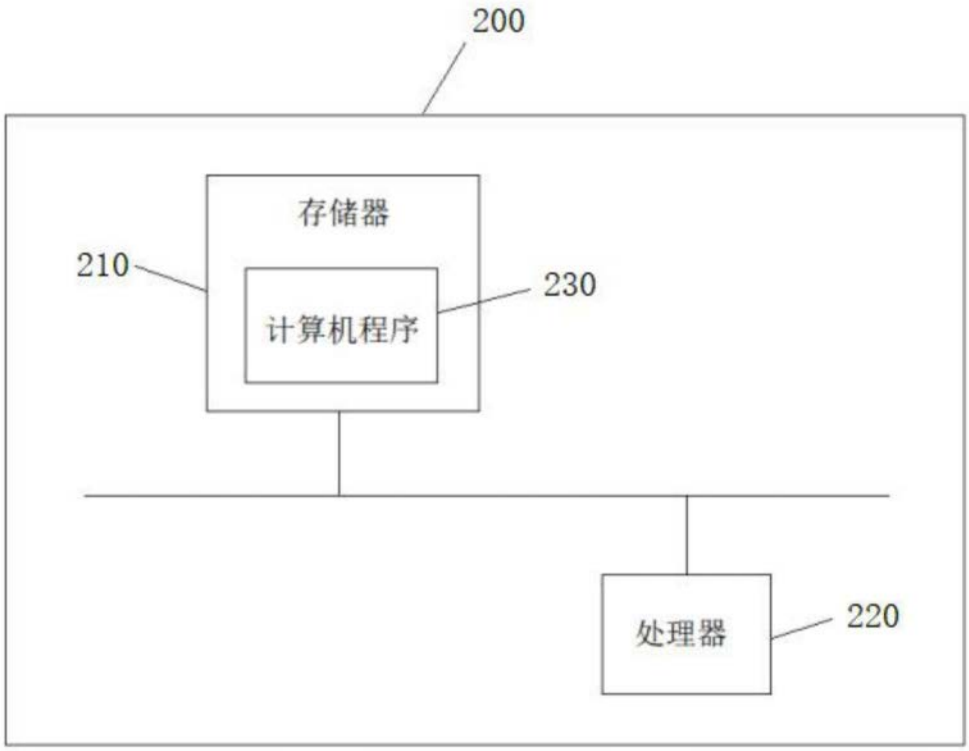



图3