(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 117891832 A
(43)申请公布日 2024.04.16

(21)申请号 202410064671.8

(22)申请日 2024.01.16

(71)申请人 北京火山引擎科技有限公司
地址 100190 北京市海淀区紫金数码园4号楼13层1309

(72)发明人 琚克俭 户蕾蕾 丁远普

(74)专利代理机构 泰和泰律师事务所 51219
专利代理师 焦玲

(51)Int.Cl.
G06F 16/245(2019.01)
G06F 16/28(2019.01)

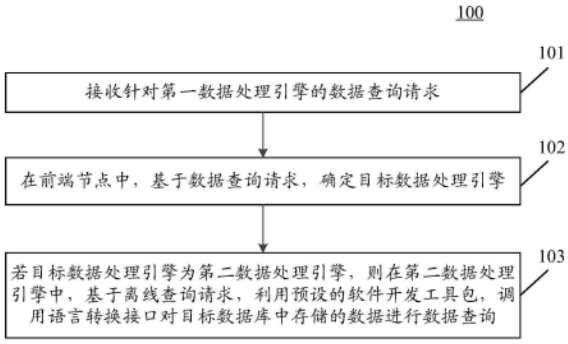
权利要求书2页 说明书12页 附图4页

(54)发明名称

数据处理方法、装置和电子设备

(57)摘要

本申请实施例公开了数据处理方法、装置和电子设备。该方法的一具体实施方式包括:接收针对第一数据处理引擎的数据查询请求;在前端节点中,基于数据查询请求,确定目标数据处理引擎,其中,第一数据处理引擎与第二数据处理引擎使用的计算机编程语言不相同;若目标数据处理引擎为第二数据处理引擎,则在第二数据处理引擎中,基于离线查询请求,利用预设的软件开发工具包,调用语言转换接口对目标数据库中存储的数据进行数据查询,语言转换接口用于接收第二语言的查询指令,对第一语言的查询指令进行调用。该实施方式通过语言转换接口解决数据查询流程在不同语言的同一套系统中的兼容问题,缓解了第一数据处理引擎在大数据量情况的读写压力。



CN 117891832 A

1.一种数据处理方法,其特征在于,包括:

接收针对第一数据处理引擎的数据查询请求,其中,所述第一数据处理引擎包括前端节点和后端节点,所述前端节点接收所述数据查询请求;

在所述前端节点中,基于所述数据查询请求,确定目标数据处理引擎,其中,所述目标数据处理引擎为所述第一数据处理引擎或者第二数据处理引擎,所述第一数据处理引擎对数据进行实时处理,所述第二数据处理引擎对数据进行离线处理,所述第一数据处理引擎与所述第二数据处理引擎使用的计算机编程语言不相同,所述第一数据处理引擎使用第一语言,所述第二数据处理引擎使用第二语言;

若所述目标数据处理引擎为所述第二数据处理引擎,则在所述第二数据处理引擎中,基于离线查询请求,利用预设的软件开发工具包,调用语言转换接口对目标数据库中存储的数据进行数据查询,其中,所述软件开发工具包用于对数据进行查询,所述语言转换接口用于接收所述第二语言的查询指令,对所述第一语言的查询指令进行调用,所述离线查询请求是基于所述数据查询请求生成的。

2.根据权利要求1所述的方法,其特征在于,在所述基于所述数据查询请求,确定目标数据处理引擎之后,所述方法还包括:

若所述目标数据处理引擎为所述第一数据处理引擎,则在所述后端节点中,基于实时查询请求,利用所述软件开发工具包对所述目标数据库中存储的数据进行数据查询,其中,所述实时查询请求是基于所述数据查询请求生成的。

3.根据权利要求2所述的方法,其特征在于,所述利用所述软件开发工具包对所述目标数据库中存储的数据进行数据查询,包括:

利用所述软件开发工具包,调用箭头格式接口对所述目标数据库中存储的数据进行数据查询,其中,所述箭头格式接口用于在列式存储格式数据与箭头格式数据之间进行转换。

4.根据权利要求1所述的方法,其特征在于,所述利用预设的软件开发工具包,调用语言转换接口对目标数据库中存储的数据进行数据查询,包括:

利用预设的软件开发工具包,调用语言转换接口,将第二语言的查询指令转换成第一语言的查询指令;

调用箭头格式接口对目标数据库中存储的数据进行数据查询,其中,所述箭头格式接口用于在行式存储格式数据与箭头格式数据之间进行转换。

5.根据权利要求1所述的方法,其特征在于,在所述前端节点中,执行如下至少一项操作:

管理所述第二数据处理引擎的元数据,其中,所述元数据为所述第二数据处理引擎可读写的数据表的元数据;

对所述第二数据处理引擎的数据拉取任务进行划分。

6.根据权利要求1所述的方法,其特征在于,所述查询指令包括以下至少一项:读取指令、写入指令和文件合并指令。

7.根据权利要求1所述的方法,其特征在于,所述基于所述数据查询请求,确定目标数据处理引擎,包括:

确定处理所述数据查询请求的耗时预测值;

响应于所述耗时预测值大于预设耗时阈值,选取第二数据处理引擎作为目标数据处理

引擎。

8.根据权利要求1所述的方法,其特征在于,所述基于所述数据查询请求,确定目标数据处理引擎,包括:

确定在处理所述数据查询请求的过程中的资源消耗预测值;

响应于所述资源消耗预测值大于预设资源消耗阈值,选取第二数据处理引擎作为目标数据处理引擎。

9.根据权利要求1所述的方法,其特征在于,所述目标接口为Java本地接口,所述第一语言为C/C++语言,所述第二语言为Java语言。

10.根据权利要求1所述的方法,其特征在于,所述第一数据引擎包括:联机分析处理引擎,所述第二数据引擎包括:分布式计算框架。

11.一种数据处理装置,其特征在于,包括:

接收单元,用于接收针对第一数据处理引擎的数据查询请求,其中,所述第一数据处理引擎包括前端节点和后端节点,所述前端节点接收所述数据查询请求;

确定单元,用于在所述前端节点中,基于所述数据查询请求,确定目标数据处理引擎,其中,所述目标数据处理引擎为所述第一数据处理引擎或者第二数据处理引擎,所述第一数据处理引擎对数据进行实时处理,所述第二数据处理引擎对数据进行离线处理,所述第一数据处理引擎与所述第二数据处理引擎使用的计算机编程语言不相同,所述第一数据处理引擎使用第一语言,所述第二数据处理引擎使用第二语言;

调用单元,用于若所述目标数据处理引擎为所述第二数据处理引擎,则在所述第二数据处理引擎中,基于离线查询请求,利用预设的软件开发工具包,调用语言转换接口对目标数据库中存储的数据进行数据查询,其中,所述软件开发工具包用于对数据进行查询,所述语言转换接口用于接收所述第二语言的查询指令,对所述第一语言的查询指令进行调用,所述离线查询请求是基于所述数据查询请求生成的。

12.一种电子设备,其特征在于,包括:

一个或多个处理器;

存储装置,其上存储有一个或多个程序,

当所述一个或多个程序被所述一个或多个处理器执行,使得所述一个或多个处理器实现如权利要求1-10中任一所述的方法。

13.一种计算机可读介质,其上存储有计算机程序,其特征在于,该程序被处理器执行时实现如权利要求1-10中任一所述的方法。

数据处理方法、装置和电子设备

技术领域

[0001] 本公开实施例涉及计算机技术领域,具体涉及数据处理方法、装置和电子设备。

背景技术

[0002] OLAP(On-line Analytical Processing,联机分析处理)数据库主要面向实时数仓的场景分析,而Spark引擎主要面向离线批量数据的处理分析。OLAP由于基于内存式的MPP(Massively Parallel Processing,即大规模并行处理)架构,所以性能比Spark引擎要高,但是在处理大批量数据的时候,OLAP会面临稳定性问题。

[0003] 伴随着OLAP数据库和Spark引擎这两套系统的集成,运维成本会比较高。同时大数据系统主要是Java语言,而OLAP系统大多都是基于C++语言,如果想同时具备两套系统的能力,即同时具备实时和离线处理的能力,则需要面临着如何解决上述两种语言的兼容性问题。

发明内容

[0004] 提供该公开内容部分以便以简要的形式介绍构思,这些构思将在后面的具体实施方式部分被详细描述。该公开内容部分并不旨在标识要求保护的技术方案的关键特征或必要特征,也不旨在用于限制所要求的保护的技术方案的范围。

[0005] 第一方面,本公开实施例提供了一种数据处理方法,包括:接收针对第一数据处理引擎的数据查询请求,其中,第一数据处理引擎包括前端节点和后端节点,前端节点接收数据查询请求;在前端节点中,基于数据查询请求,确定目标数据处理引擎,其中,目标数据处理引擎为第一数据处理引擎或者第二数据处理引擎,第一数据处理引擎对数据进行实时处理,第二数据处理引擎对数据进行离线处理,第一数据处理引擎与第二数据处理引擎使用的计算机编程语言不相同,第一数据处理引擎使用第一语言,第二数据处理引擎使用第二语言;若目标数据处理引擎为第二数据处理引擎,则在第二数据处理引擎中,基于离线查询请求,利用预设的软件开发工具包,调用语言转换接口对目标数据库中存储的数据进行数据查询,其中,软件开发工具包用于对数据进行查询,语言转换接口用于接收第二语言的查询指令,对第一语言的查询指令进行调用,离线查询请求是基于数据查询请求生成的。

[0006] 第二方面,本公开实施例提供了一种数据处理装置,包括:接收单元,用于接收针对第一数据处理引擎的数据查询请求,其中,第一数据处理引擎包括前端节点和后端节点,前端节点接收数据查询请求;确定单元,用于在前端节点中,基于数据查询请求,确定目标数据处理引擎,其中,目标数据处理引擎为第一数据处理引擎或者第二数据处理引擎,第一数据处理引擎对数据进行实时处理,第二数据处理引擎对数据进行离线处理,第一数据处理引擎与第二数据处理引擎使用的计算机编程语言不相同,第一数据处理引擎使用第一语言,第二数据处理引擎使用第二语言;调用单元,用于若目标数据处理引擎为第二数据处理引擎,则在第二数据处理引擎中,基于离线查询请求,利用预设的软件开发工具包,调用语言转换接口对目标数据库中存储的数据进行数据查询,其中,软件开发工具包用于对数据

进行查询,语言转换接口用于接收第二语言的查询指令,对第一语言的查询指令进行调用,离线查询请求是基于数据查询请求生成的。

[0007] 第三方面,本公开实施例提供了一种电子设备,包括:一个或多个处理器;存储装置,用于存储一个或多个程序,当所述一个或多个程序被所述一个或多个处理器执行,使得所述一个或多个处理器实现如第一方面所述的数据处理方法。

[0008] 第四方面,本公开实施例提供了一种计算机可读介质,其上存储有计算机程序,该程序被处理器执行时实现如第一方面所述的数据处理方法的步骤。

[0009] 本公开实施例提供的数据处理方法、装置和电子设备,通过接收针对第一数据处理引擎的数据查询请求;之后,在上述第一数据处理引擎的前端节点中,基于上述数据查询请求,确定目标数据处理引擎是上述第一数据处理引擎还是第二数据处理引擎,上述第一数据处理引擎与上述第二数据处理引擎使用的计算机编程语言不相同;若确定出上述目标数据处理引擎为上述第二数据处理引擎,则在上述第二数据处理引擎中,基于离线查询请求,利用预设的软件开发工具包,调用语言转换接口对目标数据库中存储的数据进行数据查询,上述语言转换接口用于接收第二语言的查询指令,对第一语言的查询指令进行调用。通过这种方式可以在使用不同的计算机编程语言的两个数据处理引擎进行数据处理时,通过语言转换接口解决数据查询流程在不同语言的同一套系统中的兼容问题,可以使得第二数据处理引擎执行数据读写进程,从而缓解第一数据处理引擎在大数据量情况的读写压力。

附图说明

[0010] 结合附图并参考以下具体实施方式,本公开各实施例的上述和其他特征、优点及方面将变得更加明显。贯穿附图中,相同或相似的附图标记表示相同或相似的元素。应当理解附图是示意性的,原件和元素不一定按照比例绘制。

[0011] 图1是根据本公开的数据处理方法的一个实施例的流程图;

[0012] 图2是根据本公开的数据处理方法的又一个实施例的流程图;

[0013] 图3是根据本公开的数据处理方法的一个处理方式的示意图;

[0014] 图4是根据本公开的数据处理装置的一个实施例的结构示意图;

[0015] 图5是本公开的各个实施例可以应用于其中的示例性系统架构图;

[0016] 图6是适于用来实现本公开实施例的电子设备的计算机系统的结构示意图。

具体实施方式

[0017] 下面将参照附图更详细地描述本公开的实施例。虽然附图中显示了本公开的某些实施例,然而应当理解的是,本公开可以通过各种形式来实现,而且不应该被解释为限于这里阐述的实施例,相反提供这些实施例是为了更加透彻和完整地理解本公开。应当理解的是,本公开的附图及实施例仅用于示例性作用,并非用于限制本公开的保护范围。

[0018] 应当理解,本公开的方法实施方式中记载的各个步骤可以按照不同的顺序执行,和/或并行执行。此外,方法实施方式可以包括附加的步骤和/或省略执行示出的步骤。本公开的范围在此方面不受限制。

[0019] 本文使用的术语“包括”及其变形是开放性包括,即“包括但不限于”。术语“基于”

是“至少部分地基于”。术语“一个实施例”表示“至少一个实施例”;术语“另一实施例”表示“至少一个另外的实施例”;术语“一些实施例”表示“至少一些实施例”。其他术语的相关定义将在下文描述中给出。

[0020] 需要注意,本公开中提及的“第一”、“第二”等概念仅用于对不同的装置、模块或单元进行区分,并非用于限定这些装置、模块或单元所执行的功能的顺序或者相互依存关系。

[0021] 需要注意,本公开中提及的“一个”、“多个”的修饰是示意性而非限制性的,本领域技术人员应当理解,除非在上下文另有明确指出,否则应该理解为“一个或多个”。

[0022] 本公开实施方式中的多个装置之间所交互的消息或者信息的名称仅用于说明性的目的,而并不是用于对这些消息或信息的范围进行限制。

[0023] 请参考图1,示出了根据本公开的数据处理方法的一个实施例的流程100。该数据处理方法,包括以下步骤:

[0024] 步骤101,接收针对第一数据处理引擎的数据查询请求。

[0025] 在本实施例中,数据处理方法的执行主体可以接收针对第一数据处理引擎的数据查询请求。上述数据查询请求通常为SQL(Structured Query Language,结构化查询语言)语句,SQL是具有数据操纵和数据定义等多种功能的数据库语言,这种语言具有交互性特点,能为用户提供极大的便利,数据库管理系统应充分利用SQL语言提高计算机应用系统的工作质量与效率。

[0026] 湖仓一体是一种新型的开放式架构,打通了数据仓库和数据湖,将数据仓库的高性能及管理能力与数据湖的灵活性融合了起来,底层支持多种数据类型并存,能实现数据间的相互共享,上层可以通过统一封装的接口进行访问,可同时支持实时查询和分析,为企业进行数据治理带来了更多的便利性。

[0027] 在这里,上述第一数据处理引擎通常包括前端节点(FrontEnd,FE)和后端节点(BackEnd,BE)。上述前端节点通常用于接收上述数据查询请求,除此之外,上述前端节点还可以用于管理元数据,管理客户端连接,进行查询解析规划,生成查询执行计划,查询调度(把查询下发给BE执行)等工作。上述后端节点可以用于查询执行计划的执行,副本管理等工作。即上述第一数据处理引擎通常为湖仓一体中的数据仓库。

[0028] 步骤102,在前端节点中,基于数据查询请求,确定目标数据处理引擎。

[0029] 在本实施例中,上述执行主体可以在上述第一数据处理引擎的前端节点中,基于上述数据查询请求,确定目标数据处理引擎。上述第一数据处理引擎通常对数据进行实时处理,即上述第一数据处理引擎为实时数据处理引擎。上述第二数据处理引擎通常对数据进行离线处理,即上述第二数据处理引擎为离线数据处理引擎。

[0030] 在这里,上述第二数据处理引擎通常为湖仓一体中的数据湖。

[0031] 作为示例,上述第一数据处理引擎可以具有TB(万亿字节)数据量级的数据分析能力,上述第二数据处理引擎可以具有PB数据量级及以上的数据分析能力。其中,1PB=1024TB。

[0032] 在这里,上述前端节点可以对上述数据查询请求进行分析,确定上述数据查询请求的计算逻辑是简单逻辑还是复杂逻辑。若确定出是简单逻辑,则可以选取第一数据处理引擎作为目标数据处理引擎对数据进行实时处理。若确定出是复杂逻辑,则可以选取第二数据处理引擎作为目标数据处理引擎对数据进行离线处理。

[0033] 在本实施例中,上述第一数据处理引擎与上述第二数据处理引擎使用的计算机编程语言不相同,上述第一数据处理引擎使用的计算机编程语言可以为第一语言,上述第二数据处理引擎使用的计算机编程语言可以为第二语言。

[0034] 步骤103,若目标数据处理引擎为第二数据处理引擎,则在第二数据处理引擎中,基于离线查询请求,利用预设的软件开发工具包,调用语言转换接口对目标数据库中存储的数据进行数据查询。

[0035] 在本实施例中,若上述目标数据处理引擎为第二数据处理引擎,则上述执行主体可以在上述第二数据处理引擎中,基于离线查询请求,利用预设的软件开发工具包,调用语言转换接口对目标数据库中存储的数据进行数据查询。

[0036] 在这里,上述软件开发工具包(SDK,Software Development Kit)通常用于对数据进行查询。软件开发工具包通常包括辅助开发某一类软件的相关文档、范例和工具的集合,用于实现数据查询的工具包。上述软件开发工具包通过接收上述第一数据处理引擎的后端节点发送的实时查询请求或者接收上述第二数据处理引擎的Executor执行进程发送的离线查询请求,即可实现对目标数据库的数据查询。

[0037] 上述语言转换接口用于接收第二语言(上述第二数据处理引擎使用的计算机编程语言)的查询指令,对第一语言(上述第一数据处理引擎使用的计算机编程语言)的查询指令进行调用。也就是说,上述语言转换接口可以解决两种语言的数据处理引擎之间的兼容与互操作问题。

[0038] 上述离线查询请求通常是基于上述数据查询请求生成的。上述离线查询请求通常包括用于指示查询请求是离线查询的标签。作为一种示例,上述第一数据处理引擎的前端节点可以对上述数据查询请求进行解析,可以将解析结果发送给上述第二数据处理引擎,以供上述第二数据处理引擎利用上述解析结果生成离线查询请求。作为另一种示例,上述第一数据处理引擎的前端节点可以将上述数据查询请求直接发送给上述第二数据处理引擎,上述第二数据处理引擎可以对上述数据查询请求进行解析,利用解析结果生成离线查询请求。上述解析结果可以包括上述数据查询请求所查询的数据在上述目标数据库中的存储位置。

[0039] 在这里,上述目标数据库可以为HDFS(Hadoop Distributed File System,Hadoop分布式文件系统),OSS(Object Storage Service,基于对象的存储服务)和S3(Simple Storage Service,简单存储服务)等。上述目标数据库也可以为上述第一数据处理引擎的后端节点中的数据库。

[0040] 本公开的上述实施例提供的方法通过接收针对第一数据处理引擎的数据查询请求;之后,在上述第一数据处理引擎的前端节点中,基于上述数据查询请求,确定目标数据处理引擎是上述第一数据处理引擎还是第二数据处理引擎,上述第一数据处理引擎与上述第二数据处理引擎使用的计算机编程语言不相同;若确定出上述目标数据处理引擎为上述第二数据处理引擎,则在上述第二数据处理引擎中,基于离线查询请求,利用预设的软件开发工具包,调用语言转换接口对目标数据库中存储的数据进行数据查询,上述语言转换接口用于接收第二语言的查询指令,对第一语言的查询指令进行调用。通过这种方式可以在使用不同的计算机编程语言的两个数据处理引擎进行数据处理时,通过语言转换接口解决数据查询流程在不同语言的同一套系统中的兼容问题,可以使得第二数据处理引擎执行数

据读写进程,从而缓解第一数据处理引擎在大数据量情况的读写压力。

[0041] 在一些可选的实现方式中,上述执行主体可以通过如下方式利用预设的软件开发工具包,调用语言转换接口对目标数据库中存储的数据进行数据查询:上述执行主体可以首先利用预设的软件开发工具包,调用语言转换接口,将第二语言的查询指令转换成第一语言的查询指令;之后,可以调用箭头格式(Arrow Format)接口对目标数据库中存储的数据进行数据查询。上述箭头格式接口用于在行式存储格式数据与箭头格式数据之间进行转换。Arrow Format包括一个与语言无关的内存数据结构规范、元数据序列化以及用于序列化和通用数据传输的协议。

[0042] 由于第一数据处理引擎(例如,OLAP系统)通常采用列式存储格式存储数据,而第二数据处理引擎通常采用行式存储格式存储数据,但是,列式存储格式的数据与行式存储格式的数据不能直接进行格式转换,此时,利用Arrow Format作为兼容格式,列式存储格式的数据与Arrow Format格式数据可以相互转换,行式存储格式的数据与Arrow Format格式数据也可以相互转换,由此,可以实现列式存储格式的数据与行式存储格式的数据的交互。Arrow Format作为在不同系统(引擎)中处理时的兼容格式,本身是内存式的,所以性能开销会比较小。

[0043] 在一些可选的实现方式中,上述前端节点可以管理上述第二数据处理引擎的元数据,上述元数据可以为上述第二数据处理引擎可读写的数据表的元数据。数据表的元数据是指关于表数据的数据,从表定义和字段属性到表的关系和权限,额外的元数据包括数据分析和统计信息,如索引和分区等。通过数据表的元数据可以获取数据表的存储位置,也可以获取数据表的权限,即哪些账户具有查看该数据表中数据的权限,哪些账户不具有查看该数据表中数据的权限。

[0044] 上述前端节点还可以对上述第二数据处理引擎的数据拉取任务进行划分,将划分结果发送给上述第二数据处理引擎,以供上述第二数据处理引擎的Executor执行进程对目标数据库中的数据进行拉取。在这里,上述目标数据库中的数据可以按照数据分片(Tablet)进行存储,上述元数据包括数据分片的元数据,例如,数据分片的索引、位置等。上述第一数据处理引擎的前端节点可以按照数据分片,生成至少一个数据拉取任务,在这里,一个数据拉取任务可以拉取至少一个数据分片对应的数据,作为示例,一个数据拉取任务可以拉取三个数据分片对应的数据。之后,上述第二数据处理引擎的Executor执行进程可以执行上述至少一个数据拉取任务,以从上述目标数据库进行数据拉取,从而对拉取到的数据进行数据查询。

[0045] 通过这种方式可以在第一数据处理引擎的前端节点执行第二数据处理引擎的元数据管理和任务拆分等操作,不会依赖第一数据处理引擎的后端节点。

[0046] 在一些可选的实现方式中,上述查询指令可以包括以下至少一项:读取(Writer)指令、写入(Reader)指令和文件合并(Compact)指令。第一数据处理引擎在面对小文件过多时需要有Compact操作,可以把这部分内容抽取出来单独在第二数据处理引擎执行,第二数据处理引擎可以在每条查询结束后释放资源。通过这种方式可以将第一数据处理引擎(例如,OLAP系统)中的Writer方法、Reader方法和Compact方法进行抽象剥离,将这些方法提供出去,以利用第二语言的读取指令、写入指令和文件合并指令,实现对第一语言的读取指令、写入指令和文件合并指令相应地调用,从而使用第二语言的查询指令即可调用原有的

第一语言的查询指令,无需针对第二语言重写查询指令,从而可以节约资源,此外,第二数据处理引擎可以承担更多的数据处理任务,进一步减少了第一数据处理引擎的处理压力。

[0047] 在一些可选的实现方式中,上述执行主体可以通过如下方式基于上述数据查询请求,确定目标数据处理引擎:在上述前端节点中,可以确定处理上述数据查询请求的耗时预测值。上述前端节点可以确定上述耗时预测值是否大于预设耗时阈值。上述前端节点可以将上述耗时预测值与预设耗时阈值进行比较,若确定出上述耗时预测值大于预设耗时阈值,则上述前端节点可以选取上述第二数据处理引擎作为目标数据处理引擎。若确定出上述耗时预测值小于等于预设耗时阈值,则上述前端节点可以选取上述第一数据处理引擎作为目标数据处理引擎。通过这种方式可以更加合理地选取出相应的数据处理引擎进行数据处理。

[0048] 在一些可选的实现方式中,上述执行主体可以通过如下方式基于上述数据查询请求,确定目标数据处理引擎:在上述前端节点中,可以确定在上述处理数据查询请求的过程中的资源消耗预测值。资源可以包括但不限于以下至少一项:CPU、内存、网络、I/O。上述前端节点可以确定上述资源消耗预测值是否大于预设资源消耗阈值。上述前端节点可以将上述资源消耗预测值与预设资源消耗阈值进行比较,若确定出上述资源消耗预测值大于预设资源消耗阈值,则上述前端节点可以选取上述第二数据处理引擎作为目标数据处理引擎。若确定出上述资源消耗预测值小于等于预设资源消耗阈值,则上述前端节点可以选取上述第一数据处理引擎作为目标数据处理引擎。通过这种方式可以更加合理地选取出相应的数据处理引擎进行数据处理。

[0049] 在一些可选的实现方式中,上述目标接口可以为Java本地接口(Java Native Interface,JNI),上述第一语言为C/C++语言,上述第二语言为Java语言。JNI是Java平台中的一个强大特性。应用程序可以通过JNI把C/C++代码集成进Java程序中。通过JNI,开发者在利用Java平台强大功能的同时,又不必放弃对原有代码的投资;因为JNI是Java平台定义的规范接口,当程序员向Java代码集成本地库时,只要在一个平台中解决了语言互操作问题,就可以把该解决方案比较容易的移植到其他Java平台中。当Java平台部署到本地系统中,有必要做到让Java程序与本地代码协同工作。部分是由于遗留代码(保护原有的投资)的问题(一些效率敏感的代码用C实现,但现在Java VM的执行效率完全可信赖),工程师们很早就开始以C/C++为基础构建Java应用,所以,C/C++代码将长时间的与Java应用共存。JNI在利用强大Java平台的同时,仍然可以用其他语言写程序。JNI是一套双向的接口,允许Java与本地代码间的互操作。

[0050] 在一些可选的实现方式中,上述第一数据引擎包括:联机分析处理引擎,OLAP是一种用于分析和查询大规模数据集的计算机处理技术。OLAP技术主要用于多维数据分析和数据挖掘,通过提供多维数据模型和多维查询功能,帮助用户从不同角度和层次上对数据进行分析和查询,侧重分析决策。

[0051] 作为示例,上述第一数据库可以包括但不限于:StarRocks、Doris和ClickHouse。StarRocks是新一代极速全场景MPP数据库,StarRocks的愿景是能够让用户的数据分析变得更加简单和敏捷。用户无需经过复杂的预处理,就可以用StarRocks来支持多种数据分析场景的极速分析。StarRocks架构简洁,采用了全面向量化引擎,并配备全新设计的CBO(Cost Based Optimizer)优化器,查询速度(尤其是多表关联查询)远超同类产品。

StarRocks能很好地支持实时数据分析,并能实现对实时更新数据的高效查询。StarRocks还支持现代化物化视图,进一步加速查询。使用StarRocks,用户可以灵活构建包括大宽表、星型模型、雪花模型在内的各类模型。StarRocks兼容MySQL(开源的关系型数据库管理系统)协议,支持标准SQL语法,易于对接使用,全系统无外部依赖,高可用,易于运维管理。

[0052] Doris也是一个MPP架构的、专门用于OLAP查询的数据库产品。Doris不仅可以满足多种数据分析需求,如固定历史报表,实时数据分析,交互式数据分析,探索式数据分析,而且集群扩展能力非常突出,可以支持10PB以上的超大数据集。另外,Doris对实时数据的支持也非常强大,因此是一款综合能力非常强的数据库。

[0053] ClickHouse是一款基于MPP架构的列式存储数据库,能够使用SQL查询语句实时生成数据分析结果。ClickHouse的全称是Click Stream,Data WareHouse。ClickHouse也是第一款实现向量化查询引擎的开源数据库,也是一款专注于OLAP查询的数据库。ClickHouse直接将OLAP查询的耗时压缩到了压秒级。

[0054] 上述第二数据处理引擎可以包括:分布式计算框架。作为示例,上述第二数据处理引擎可以为Spark,Spark是专为大规模数据分析处理而设计的开源分布式计算框架。使用内存计算技术和有向无环图(Directed Acyclic Graph,DAG)提供比MapReduce引擎更快的分析处理能力。Spark具有PB级及以上的离线分析能力。

[0055] 由于OLAP引擎采用MPP架构,适用于实时数仓场景,但当数据量达到PB数据量级时,基本上不能采用OLAP引擎进行数据处理,原因是内存不足,时间运行较长,且当执行的任务有失败时,没有重试机制。这时便可以利用Spark解决这样的问题。因此将OLAP引擎和Spark结合可以满足:实时性(TB数据量级的数据实时分析能力)和大数据量(PB数据量级及以上的离线分析能力)。

[0056] 继续参考图2,其示出了数据处理方法的又一个实施例的流程200。该数据处理方法的流程200,包括以下步骤:

[0057] 步骤201,接收针对第一数据处理引擎的数据查询请求。

[0058] 步骤202,在前端节点中,基于数据查询请求,确定目标数据处理引擎。

[0059] 步骤203,若目标数据处理引擎为第二数据处理引擎,则在第二数据处理引擎中,基于离线查询请求,利用预设的软件开发工具包,调用语言转换接口对目标数据库中存储的数据进行数据查询。

[0060] 在本实施例中,步骤201-203可以按照与步骤101-103类似的方式执行,在此不再赘述。

[0061] 步骤204,若目标数据处理引擎为第一数据处理引擎,则在后端节点中,基于实时查询请求,利用软件开发工具包对目标数据库中存储的数据进行数据查询。

[0062] 在本实施例中,若上述目标数据处理引擎为第一数据处理引擎,则上述执行主体可以在上述第一数据处理引擎的后端节点中,基于实时查询请求,利用上述软件开发工具包对上述目标数据库中存储的数据进行数据查询。

[0063] 在这里,上述实时查询请求是基于上述数据查询请求生成的,上述实时查询请求通常包括用于指示查询请求是实时查询的标签。上述第一数据处理引擎的前端节点可以对上述数据查询请求进行解析,得到解析结果,之后,可以将上述解析结果发送给上述后端节点,上述后端节点可以利用上述解析结果生成实时查询请求,而后,可以将该实时查询请求

发送给上述软件开发工具包,从而利用上述软件开发工具包实现数据查询。

[0064] 从图2中可以看出,与图1对应的实施例相比,本实施例中的数据处理方法的流程200体现了若目标数据处理引擎为第一数据处理引擎,则在第一数据处理引擎的后端节点中,基于实时查询请求,利用软件开发工具包对目标数据库中存储的数据进行数据查询的步骤。由此,本实施例描述的方案可以通过统一的SDK实现实时离线一体化的数据读写。

[0065] 在一些可选的实现方式中,上述执行主体可以通过如下方式利用上述软件开发工具包对上述目标数据库中存储的数据进行数据查询:上述执行主体可以利用上述软件开发工具包,调用箭头格式接口对上述目标数据库中存储的数据进行数据查询。上述箭头格式接口可以用于在列式存储格式数据与箭头格式数据之间进行转换。Arrow Format包括一个与语言无关的内存数据结构规范、元数据序列化以及用于序列化和通用数据传输的协议。

[0066] 由于第一数据处理引擎通常采用列式存储格式存储数据,而第二数据处理引擎通常采用行式存储格式存储数据,但是,列式存储格式的数据与行式存储格式的数据不能直接进行格式转换,此时,利用Arrow Format作为兼容格式,列式存储格式的数据与Arrow Format格式数据可以相互转换,行式存储格式的数据与Arrow Format格式数据也可以相互转换,由此,可以实现列式存储格式的数据与行式存储格式的数据的交互。Arrow Format作为在不同系统(引擎)中处理时的兼容格式,本身是内存式的,所以性能开销会比较小。

[0067] 继续参见图3,图3是根据本实施例的数据处理方法的一个处理方式的示意图。在图3中,客户端会向OLAP引擎发送SQL语句,OLAP引擎包括FE节点和BE节点,FE节点会对接收到的SQL语句进行解析,确定SQL语句的查询逻辑是简单查询还是复杂查询。若确定出是简单查询,则OLAP引擎的BE节点会向统一SDK发送实时查询请求,统一SDK会在目标数据库中对实时查询请求所请求的数据进行数据查询。统一SDK会兼容实时数据查询和离线数据查询。

[0068] 若确定出是复杂查询,则OLAP引擎的FE节点会将解析结果发送给Spark引擎,Spark引擎的Executor执行进程会向统一SDK发送离线查询请求,统一SDK会在目标数据库中对离线查询请求所请求的数据进行数据查询。具体地,在利用统一SDK进行数据查询的过程中,会调用JNI接口,JNI接口会接收Java语言的查询指令,对C/C++代码的查询指令进行调用,从而解决数据查询流程在不同语言的同一套系统中的兼容问题。

[0069] 此外,由于OLAP引擎通常采用列式存储格式存储数据,而Spark引擎通常采用行式存储格式存储数据,但是,列式存储格式的数据与行式存储格式的数据不能直接进行格式转换,此时,利用Arrow Format作为兼容格式,列式存储格式的数据与Arrow Format格式数据可以相互转换,行式存储格式的数据与Arrow Format格式数据也可以相互转换,由此,可以实现列式存储格式的数据与行式存储格式的数据的交互,即实现了OLAP引擎与Spark引擎的数据交互。

[0070] 图3中的目标数据库可以为HDFS、OSS、TOS和S3中的任一种。

[0071] 通过图3这种处理方式实现一套数据存储,一套统一SDK,两个可选的引擎。数据格式优先选择OLAP引擎本身的数据格式,因为OLAP引擎本身的数据格式中具备了针对性的优化,例如索引,字典等各种提升性能的手段,但是为了兼容其他的开源格式能够进入系统,所以统一的SDK具备了两个能力:一是能够适配大批量查询的离线任务和高频更新写入查询的实时任务的接口切换;二是能够兼容来自不同的开源格式按照原有的开源格式继续读

写。

[0072] 进一步参考图4,作为对上述各图所示方法的实现,本申请提供了一种数据处理装置的一个实施例,该装置实施例与图1所示的方法实施例相对应,该装置具体可以应用于各种电子设备中。

[0073] 如图4所示,本实施例的数据处理装置400包括:接收单元401、确定单元402和调用单元403。其中,接收单元401用于接收针对第一数据处理引擎的数据查询请求,其中,上述第一数据处理引擎包括前端节点和后端节点,上述前端节点接收上述数据查询请求;确定单元402用于在上述前端节点中,基于上述数据查询请求,确定目标数据处理引擎,其中,上述目标数据处理引擎为上述第一数据处理引擎或者第二数据处理引擎,上述第一数据处理引擎对数据进行实时处理,上述第二数据处理引擎对数据进行离线处理,上述第一数据处理引擎与上述第二数据处理引擎使用的计算机编程语言不相同,上述第一数据处理引擎使用第一语言,上述第二数据处理引擎使用第二语言;调用单元403用于若上述目标数据处理引擎为上述第二数据处理引擎,则在上述第二数据处理引擎中,基于离线查询请求,利用预设的软件开发工具包,调用语言转换接口对目标数据库中存储的数据进行数据查询,其中,上述软件开发工具包用于对数据进行查询,上述语言转换接口用于接收第二语言的查询指令,对第一语言的查询指令进行调用,上述离线查询请求是基于上述数据查询请求生成的。

[0074] 在本实施例中,数据处理装置400的接收单元401、确定单元402和调用单元403的具体处理可以参考图1对应实施例中的步骤101、步骤102和步骤103。

[0075] 在一些可选的实现方式中,上述数据处理装置400还可以包括:查询单元(图中未示出)。上述查询单元可以用于若上述目标数据处理引擎为上述第一数据处理引擎,则在上述后端节点中,基于实时查询请求,利用上述软件开发工具包对上述目标数据库中存储的数据进行数据查询,其中,上述实时查询请求是基于上述数据查询请求生成的。

[0076] 在一些可选的实现方式中,上述查询单元进一步用于通过如下方式利用上述软件开发工具包对上述目标数据库中存储的数据进行数据查询:利用上述软件开发工具包,调用箭头格式接口对上述目标数据库中存储的数据进行数据查询,其中,上述箭头格式接口用于在列式存储格式数据与箭头格式数据之间进行转换。

[0077] 在一些可选的实现方式中,上述调用单元403进一步用于通过如下方式利用预设的软件开发工具包,调用语言转换接口对目标数据库中存储的数据进行数据查询:利用预设的软件开发工具包,调用语言转换接口,将第二语言的查询指令转换成第一语言的查询指令;调用箭头格式接口对目标数据库中存储的数据进行数据查询,其中,上述箭头格式接口用于在行式存储格式数据与箭头格式数据之间进行转换。

[0078] 在一些可选的实现方式中,在上述前端节点中,执行如下至少一项操作:管理上述第二数据处理引擎的元数据,其中,上述元数据为上述第二数据处理引擎可读写的数据表的元数据;对上述第二数据处理引擎的数据拉取任务进行划分。

[0079] 在一些可选的实现方式中,上述查询指令包括以下至少一项:读取指令、写入指令和文件合并指令。

[0080] 在一些可选的实现方式中,上述确定单元402进一步用于通过如下方式基于上述数据查询请求,确定目标数据处理引擎:确定处理上述数据查询请求的耗时预测值;响应于上述耗时预测值大于预设耗时阈值,选取第二数据处理引擎作为目标数据处理引擎。

[0081] 在一些可选的实现方式中,上述确定单元402进一步用于通过如下方式基于上述数据查询请求,确定目标数据处理引擎:确定在处理上述数据查询请求的过程中的资源消耗预测值;响应于上述资源消耗预测值大于预设资源消耗阈值,选取第二数据处理引擎作为目标数据处理引擎。

[0082] 在一些可选的实现方式中,上述目标接口为Java本地接口,上述第一语言为C/C++语言,上述第二语言为Java语言。

[0083] 在一些可选的实现方式中,上述第一数据引擎包括:联机分析处理引擎,上述第二数据引擎包括:分布式计算框架。

[0084] 图5示出了可以应用本公开的数据处理方法的实施例的示例性系统架构500。

[0085] 如图5所示,系统架构500可以包括终端设备5011、5012、5013,网络502和服务器503。网络502用以在终端设备5011、5012、5013和服务器503之间提供通信链路的介质。网络502可以包括各种连接类型,例如有线、无线通信链路或者光纤电缆等等。

[0086] 用户可以使用终端设备5011、5012、5013通过网络502与服务器503交互,以发送或接收消息等,例如,服务器503可以接收终端设备5011、5012、5013发送的针对第一数据处理引擎的数据查询请求。终端设备5011、5012、5013上可以安装有各种通讯客户端应用,例如短视频软件、搜索引擎等。

[0087] 终端设备5011、5012、5013可以是硬件,也可以是软件。当终端设备5011、5012、5013为硬件时,可以是具有显示屏并且支持信息交互的各种电子设备,包括但不限于智能手机、平板电脑、膝上型便携计算机等。当终端设备5011、5012、5013为软件时,可以安装在上述所列举的电子设备中。其可以实现成多个软件或软件模块(例如用来提供分布式服务的多个软件或软件模块),也可以实现成单个软件或软件模块。在此不做具体限定。

[0088] 服务器503可以是提供各种服务的服务器。例如,可以是对数据查询请求进行处理的后台服务器。服务器503可以接收针对第一数据处理引擎的数据查询请求,其中,上述第一数据处理引擎包括前端节点和后端节点,上述前端节点接收上述数据查询请求;之后,可以在上述前端节点中,基于上述数据查询请求,确定目标数据处理引擎,其中,上述目标数据处理引擎为上述第一数据处理引擎或者第二数据处理引擎,上述第一数据处理引擎对数据进行实时处理,上述第二数据处理引擎对数据进行离线处理,上述第一数据处理引擎与上述第二数据处理引擎使用的计算机编程语言不相同,上述第一数据处理引擎使用第一语言,上述第二数据处理引擎使用第二语言;若上述目标数据处理引擎为上述第二数据处理引擎,则在上述第二数据处理引擎中,基于离线查询请求,利用预设的软件开发工具包,调用语言转换接口对目标数据库中存储的数据进行数据查询,其中,上述软件开发工具包用于对数据进行查询,上述语言转换接口用于接收第二语言的查询指令,对第一语言的查询指令进行调用,上述离线查询请求是基于上述数据查询请求生成的。

[0089] 需要说明的是,服务器503可以是硬件,也可以是软件。当服务器503为硬件时,可以实现成多个服务器组成的分布式服务器集群,也可以实现成单个服务器。当服务器503为软件时,可以实现成多个软件或软件模块(例如用来提供分布式服务),也可以实现成单个软件或软件模块。在此不做具体限定。

[0090] 还需要说明的是,本公开实施例所提供的数据处理方法通常由服务器503执行,则数据处理装置通常设置于服务器503中。

[0091] 应该理解,图5中的终端设备、网络和服务器的数目仅仅是示意性的。根据实现需要,可以具有任意数目的终端设备、网络和服务器。

[0092] 下面参考图6,其示出了适于用来实现本公开的实施例的电子设备(例如图5中的服务器)600的结构示意图。图6示出的电子设备仅仅是一个示例,不应对本公开的实施例的功能和使用范围带来任何限制。

[0093] 如图6所示,电子设备600可以包括处理装置(例如中央处理器、图形处理器等)601,其可以根据存储在只读存储器(ROM)602中的程序或者从存储装置608加载到随机访问存储器(RAM)603中的程序而执行各种适当的动作和处理。在RAM 603中,还存储有电子设备600操作所需的各种程序和数据。处理装置601、ROM 602以及RAM 603通过总线604彼此相连。输入/输出(I/O)接口605也连接至总线604。

[0094] 通常,以下装置可以连接至I/O接口605:包括例如触摸屏、触摸板、键盘、鼠标、摄像头、麦克风、加速度计、陀螺仪等的输入装置606;包括例如液晶显示器(LCD)、扬声器、振动器等的输出装置607;包括例如磁带、硬盘等的存储装置608;以及通信装置609。通信装置609可以允许电子设备600与其他设备进行无线或有线通信以交换数据。虽然图6示出了具有各种装置的电子设备600,但是应理解的是,并不要求实施或具备所有示出的装置。可以替代地实施或具备更多或更少的装置。图6中示出的每个方框可以代表一个装置,也可以根据需要代表多个装置。

[0095] 特别地,根据本公开的实施例,上文参考流程图描述的过程可以被实现为计算机软件程序。例如,本公开的实施例包括一种计算机程序产品,其包括承载在计算机可读介质上的计算机程序,该计算机程序包含用于执行流程图所示的方法的程序代码。在这样的实施例中,该计算机程序可以通过通信装置609从网络上被下载和安装,或者从存储装置608被安装,或者从ROM 602被安装。在该计算机程序被处理装置601执行时,执行本公开的实施例的方法中限定的上述功能。需要说明的是,本公开的实施例所述的计算机可读介质可以是计算机可读信号介质或者计算机可读存储介质或者是上述两者的任意组合。计算机可读存储介质例如可以是——但不限于——电、磁、光、电磁、红外线、或半导体的系统、装置或器件,或者任意以上的组合。计算机可读存储介质的更具体的例子可以包括但不限于:具有一个或多个导线的电连接、便携式计算机磁盘、硬盘、随机访问存储器(RAM)、只读存储器(ROM)、可擦式可编程只读存储器(EPROM或闪存)、光纤、便携式紧凑磁盘只读存储器(CD-ROM)、光存储器件、磁存储器件、或者上述的任意合适的组合。在本公开的实施例中,计算机可读存储介质可以是任何包含或存储程序的有形介质,该程序可以被指令执行系统、装置或者器件使用或者与其结合使用。而在本公开的实施例中,计算机可读信号介质可以包括在基带中或者作为载波一部分传播的数据信号,其中承载了计算机可读的程序代码。这种传播的数据信号可以采用多种形式,包括但不限于电磁信号、光信号或上述的任意合适的组合。计算机可读信号介质还可以是计算机可读存储介质以外的任何计算机可读介质,该计算机可读信号介质可以发送、传播或者传输用于由指令执行系统、装置或者器件使用或者与其结合使用的程序。计算机可读介质上包含的程序代码可以用任何适当的介质传输,包括但不限于:电线、光缆、RF(射频)等等,或者上述的任意合适的组合。

[0096] 上述计算机可读介质可以是上述电子设备中所包含的;也可以是单独存在,而未装配入该电子设备中。上述计算机可读介质承载有一个或者多个程序,当上述一个或者多

个程序被该电子设备执行时,使得该电子设备:接收针对第一数据处理引擎的数据查询请求,其中,上述第一数据处理引擎包括前端节点和后端节点,上述前端节点接收上述数据查询请求;在上述前端节点中,基于上述数据查询请求,确定目标数据处理引擎,其中,上述目标数据处理引擎为上述第一数据处理引擎或者第二数据处理引擎,上述第一数据处理引擎对数据进行实时处理,上述第二数据处理引擎对数据进行离线处理,上述第一数据处理引擎与上述第二数据处理引擎使用的计算机编程语言不相同,上述第一数据处理引擎使用第一语言,上述第二数据处理引擎使用第二语言;若上述目标数据处理引擎为上述第二数据处理引擎,则在上述第二数据处理引擎中,基于离线查询请求,利用预设的软件开发工具包,调用语言转换接口对目标数据库中存储的数据进行数据查询,其中,上述软件开发工具包用于对数据进行查询,上述语言转换接口用于接收第二语言的查询指令,对第一语言的查询指令进行调用,上述离线查询请求是基于上述数据查询请求生成的。

[0097] 可以以一种或多种程序设计语言或其组合来编写用于执行本公开的实施例的操作的计算机程序代码,所述程序设计语言包括面向对象的程序设计语言—诸如Java、Smalltalk、C++,还包括常规的过程式程序设计语言—诸如“C”语言或类似的程序设计语言。程序代码可以完全地在用户计算机上执行、部分地在用户计算机上执行、作为一个独立的软件包执行、部分在用户计算机上部分在远程计算机上执行、或者完全在远程计算机或服务器上执行。在涉及远程计算机的情形中,远程计算机可以通过任意种类的网络——包括局域网(LAN)或广域网(WAN)——连接到用户计算机,或者,可以连接到外部计算机(例如利用因特网服务提供商来通过因特网连接)。

[0098] 附图中的流程图和框图,图示了按照本公开各种实施例的系统、方法和计算机程序产品的可能实现的体系架构、功能和操作。在这点上,流程图或框图中的每个方框可以代表一个模块、程序段、或代码的一部分,该模块、程序段、或代码的一部分包含一个或多个用于实现规定的逻辑功能的可执行指令。也应当注意,在有些作为替换的实现中,方框中所标注的功能也可以以不同于附图中所标注的顺序发生。例如,两个接连地表示的方框实际上可以基本并行地执行,它们有时也可以按相反的顺序执行,这依所涉及的功能而定。也要注意的是,框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合,可以用执行规定的功能或操作的专用的基于硬件的系统来实现,或者可以用专用硬件与计算机指令的组合来实现。

[0099] 描述于本公开的实施例中所涉及到的单元可以通过软件的方式实现,也可以通过硬件的方式来实现。所描述的单元也可以设置在处理器中,例如,可以描述为:一种处理器包括接收单元、确定单元和调用单元。其中,这些单元的名称在某种情况下并不构成对该单元本身的限定,例如,接收单元还可以被描述为“接收针对第一数据处理引擎的数据查询请求的单元”。

[0100] 以上描述仅为本公开的较佳实施例以及对所运用技术原理的说明。本领域技术人员应当理解,本公开的实施例中所涉及的发明范围,并不限于上述技术特征的特定组合而成的技术方案,同时也应涵盖在不脱离上述发明构思的情况下,由上述技术特征或其等同特征进行任意组合而形成的其它技术方案。例如上述特征与本公开的实施例中公开的(但不限于)具有类似功能的技术特征进行互相替换而形成的技术方案。

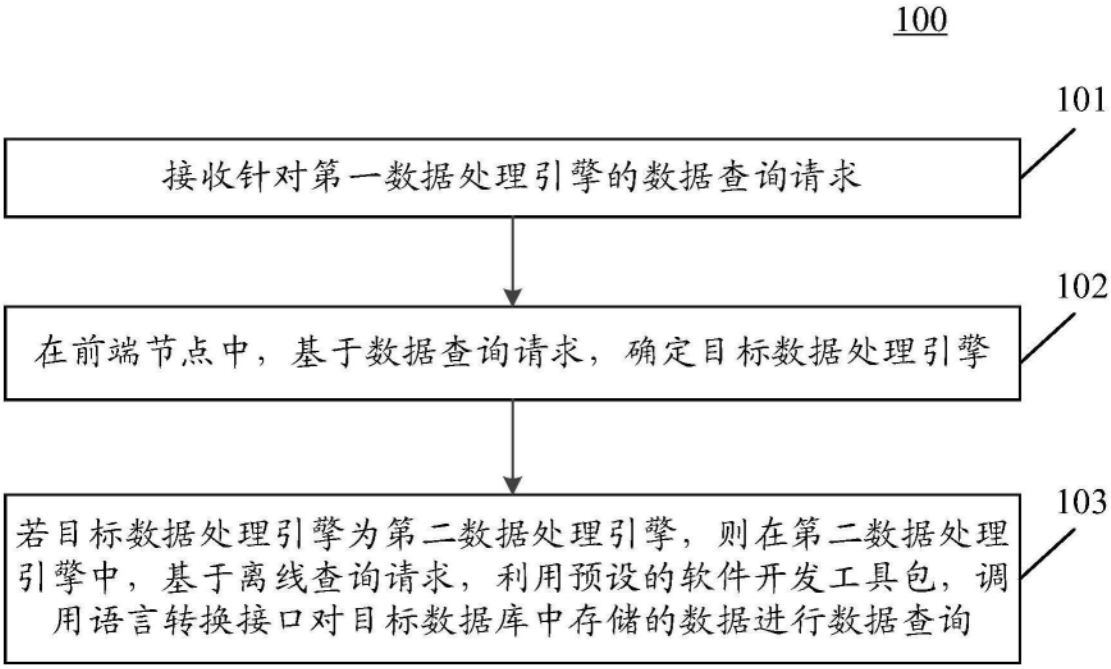

100
101
接收针对第一数据处理引擎的数据查询请求
102
在前端节点中，基于数据查询请求，确定目标数据处理引擎
103
若目标数据处理引擎为第二数据处理引擎，则在第二数据处理引擎中，基于离线查询请求，利用预设的软件开发工具包，调用语言转换接口对目标数据库中存储的数据进行数据查询

图1

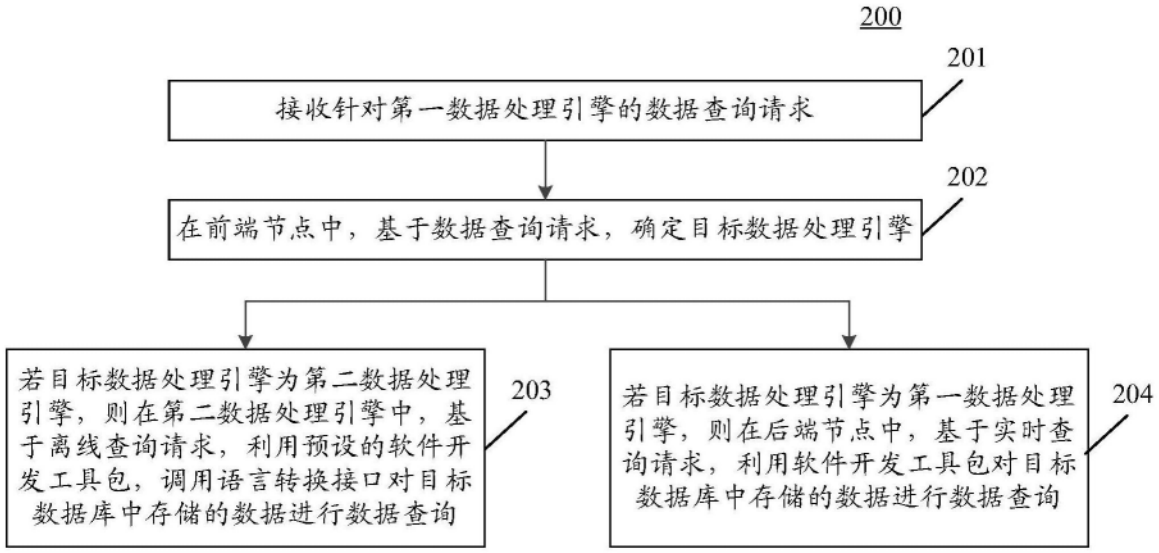

200
201
接收针对第一数据处理引擎的数据查询请求
202
在前端节点中，基于数据查询请求，确定目标数据处理引擎
若目标数据处理引擎为第二数据处理引擎，则在第二数据处理引擎中，基于离线查询请求，利用预设的软件开发工具包，调用语言转换接口对目标数据库中存储的数据进行数据查询
203
若目标数据处理引擎为第一数据处理引擎，则在后端节点中，基于实时查询请求，利用软件开发工具包对目标数据库中存储的数据进行数据查询
204

图2

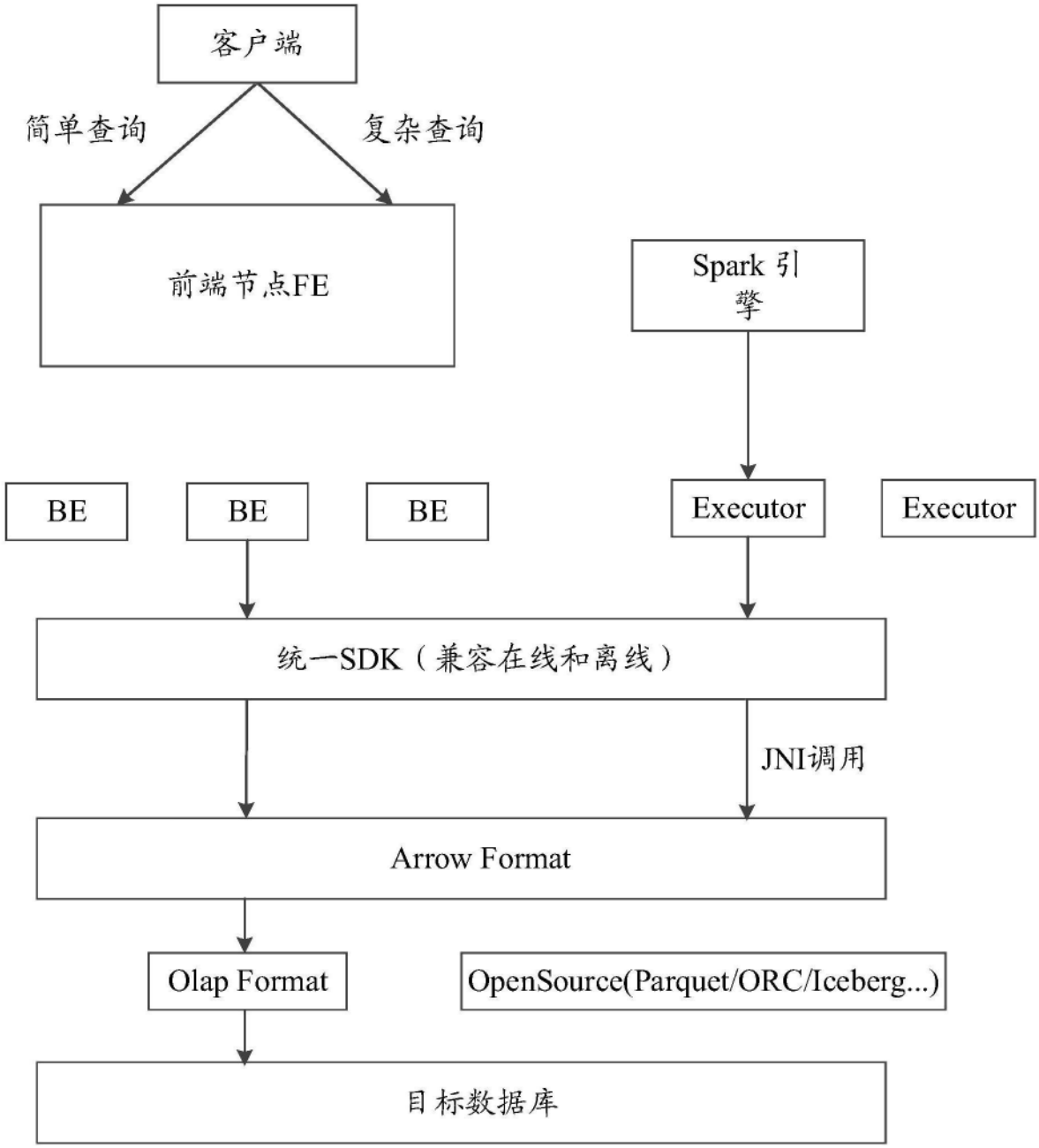

客户端
简单查询
复杂查询
前端节点FE
Spark 引擎
BE
BE
BE
Executor
Executor
统一SDK（兼容在线和离线）
JNI调用
Arrow Format
Olap Format
OpenSource(Parquet/ORC/Iceberg...)
目标数据库

图3

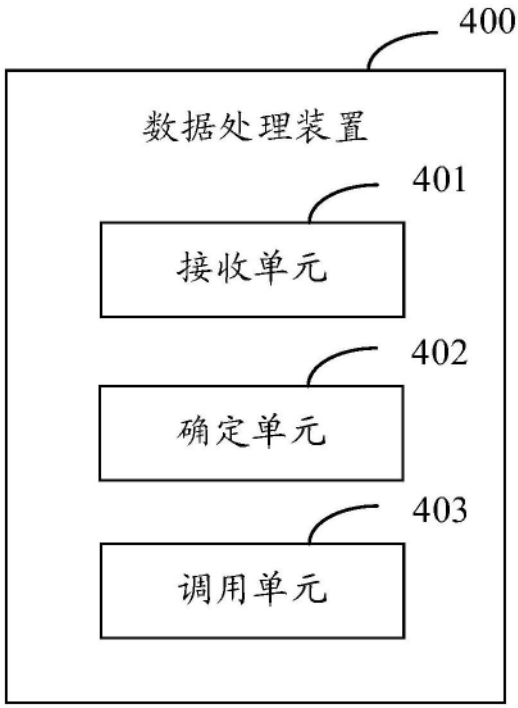

400
数据处理装置
401
接收单元
402
确定单元
403
调用单元

图4

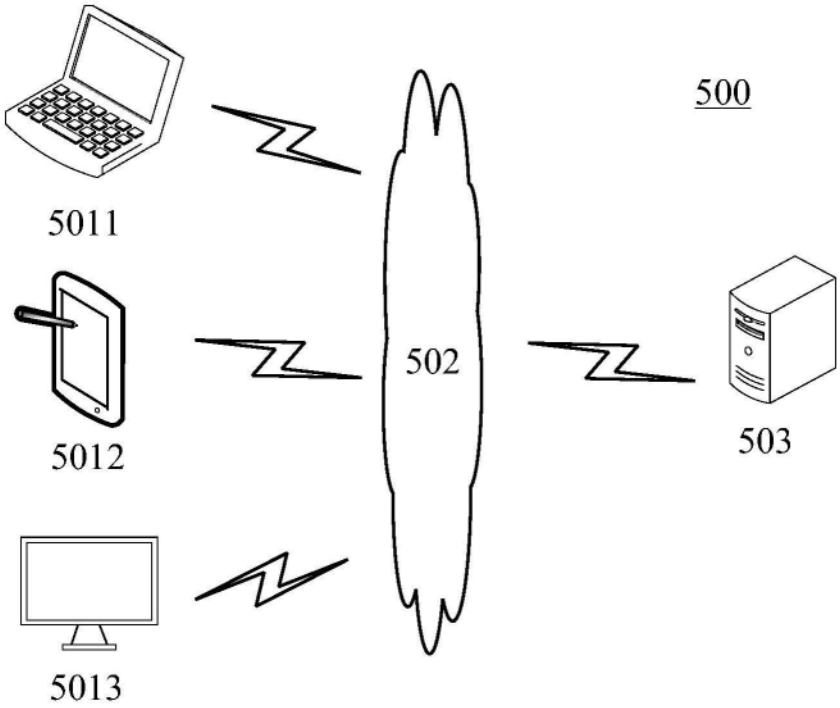

500
5011
5012
5013
502
503

图5

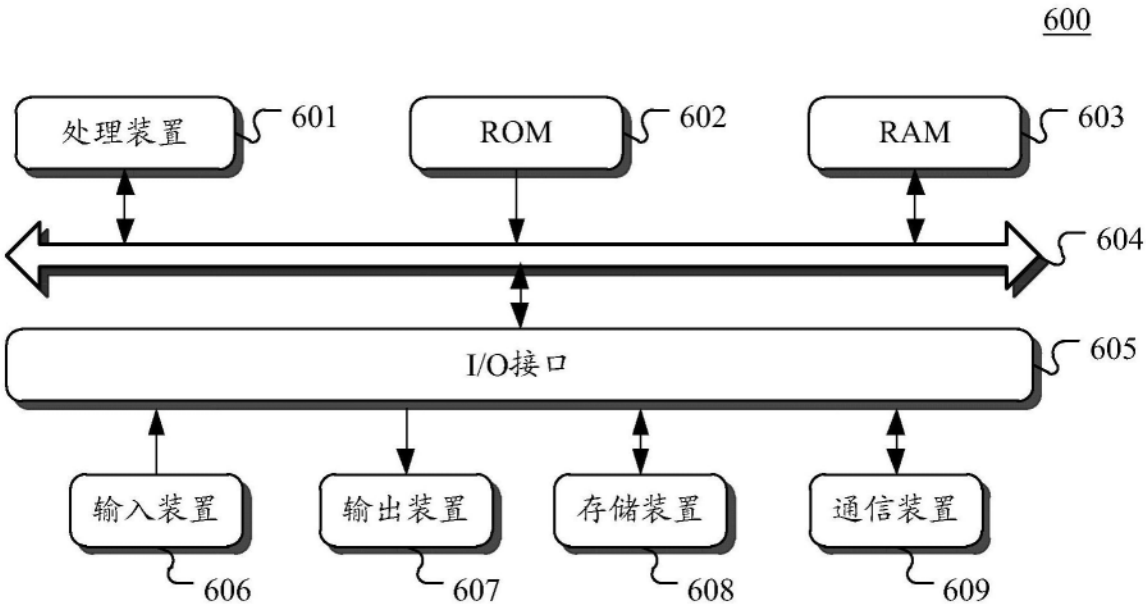

600
处理装置
601
ROM
602
RAM
603
604
I/O接口
605
输入装置
606
输出装置
607
存储装置
608
通信装置
609

图6