阿里云
大数据轻量专有云

技术白皮书

产品版本：V1.1.0

文档版本：20180327

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止**： 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告**： 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明**： 您也可以通过按**Ctrl** + **A**选中全部文件。
>	多级菜单递进。	**设置** > **网络** > **设置网络类型**
粗体	表示按键、菜单、页面名称等UI元素。	单击 **确定**。
courier字体	命令。	执行 cd /d C:/windows 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	bae log list --instanceid *Instance_ID*
[]或者[a|b]	表示可选项，至多选择一个。	ipconfig *[-all|-t]*
{}或者{a|b}	表示必选项，至多选择一个。	swich *{stand | slave}*

目录

1 产品概述

大数据轻量专有云（Apsara Stack Insight）是阿里巴巴基于阿里云飞天分布式操作系统研发的大数据产品集合，是阿里云大数据产品在小规模集群上应用的解决方案。

大数据轻量专有云包括大数据计算服务MaxCompute、分析型数据库AnalyticDB、大数据开发套件DataWorks、大数据应用加速器和关系网络分析Graph Analytics 五款核心产品。

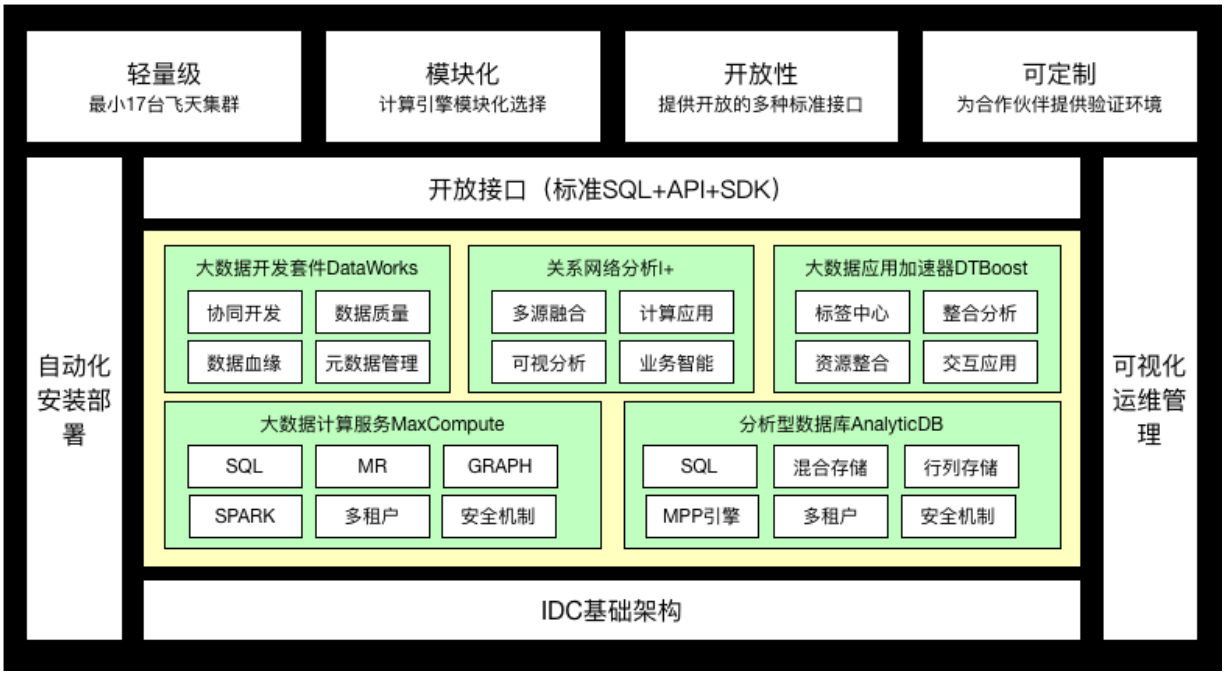
大数据轻量专有云极大的降低了客户使用阿里云大数据产品的成本和门槛，这将使阿里云大数据产品普惠到各行各业，尤其是市级、地区级的单位和企业。

2 系统架构

2.1 产品架构

大数据轻量专有云是基于标准的IDC基础架构、x86服务器、以太网交换机的大数据产品。

大数据轻量专有云的产品架构如下图所示。



在硬件服务器之上，大数据轻量专有云搭载飞天操作系统，形成大规模集群。大数据计算服务MaxCompute和分析型数据库AnalyticDB是基于飞天操作系统开发的大数据计算平台。在计算平台之上，大数据轻量专有云提供大数据开发套件DataWorks、关系网络分析I+ 和大数据应用加速器DTBoost。

大数据轻量专有云支持自动化安装部署和可视化运维。运维人员可通过工具便捷的安装部署大数据轻量专有云，并可通过可视化运维管理工具维护大数据轻量专有云中的各产品，极大的提升运维人员的工作效率。

大数据轻量专有云主要特点如下：

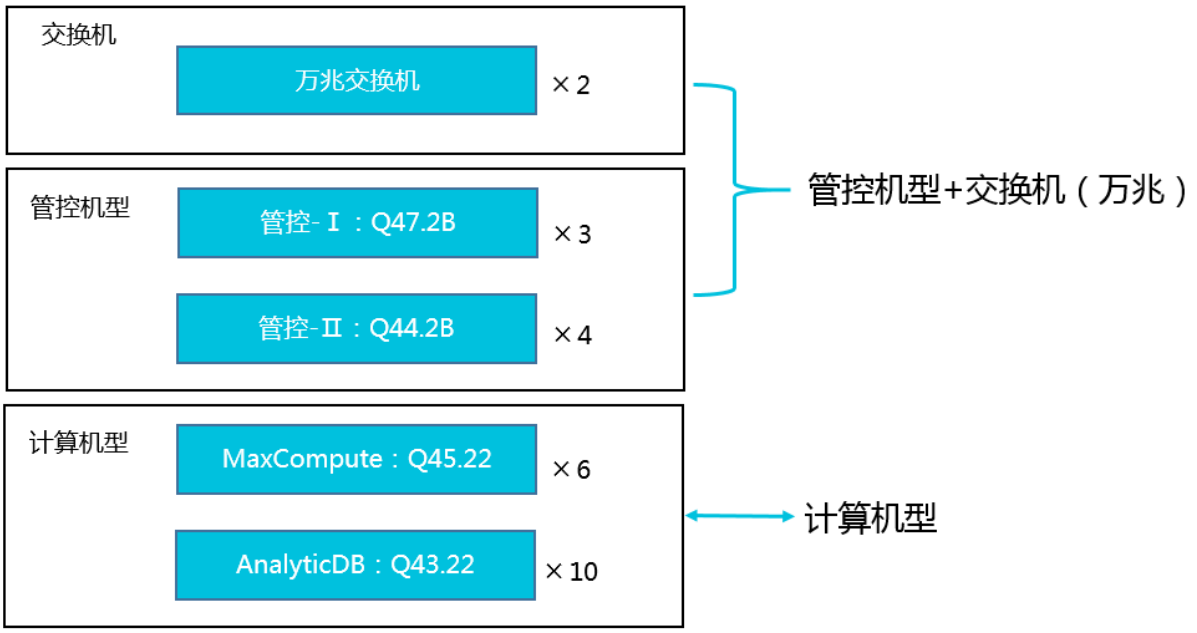
- 轻量化：支持15台~96台服务器的飞天集群。
- 模块化：支持模块化组合交付，常用组合如下：
  - Maxcompute与DataWorks组合
  - Maxcompute、AnalyticDB与DataWorks组合

  - AnalyticDB
  - AnalyticDB与I + 组合
  - Maxcompute 、DataWorks、AnalyticDB、I + 与DTBoost组合
- 开放性：提供多种标准接口。
- 可定制：为合作伙伴提供预验证的环境。

2.2 交付形态

大数据轻量专有云对硬件的要求，以及计算机型对服务器的要求。

大数据轻量专有云的交付形态如下图所示。



硬件要求：至少包含2台万兆交换机、7台管控服务器。DataWorks 、I+和 DTBoost 软件部署在7台管控服务器中，不需要额外的服务器。

计算机型对服务器要求：MaxCompute至少需要6台物理服务器，AnalyticDB至少需要10台服务器。

3 系统容量

介绍大数据轻量专有云中AnalyticDB和MaxCompute的性能参数。

AnalyticDB性能参数

类型	机器数（台）	最大数据量（TB）	查询（多维）	查询（复杂度）	写入（rec/s）	写入（MB/s)
高性能	10~30	10~50	200~1000并发	70~350并发	8万~40万	30~150
大存储	10~30	100~500	20~100并发	10~50并发	8万~40万	30~150

MaxCompute性能参数

机型	机器数（台）	理论最大数据量	单机磁盘利用率	压缩比
Q45.2B	6~30	336.6TB~1PB	0.668	3

4 产品优势

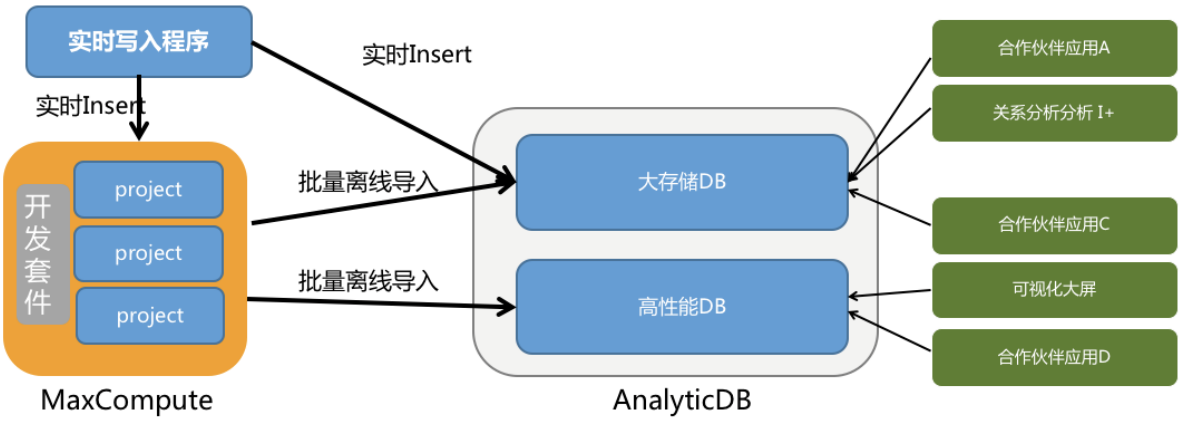
大数据轻量专有云的主要优势如下：

- 大数据轻量专有云的集群规模小，输出的产品能力强，能够形成数据链路闭环。
- MaxCompute能够高效的处理海量数据的离线分析，在阿里巴巴集团内稳定运行。
- MaxCompute提供多种授权管理手段，包括ACL、角色授权、Policy授权、跨Project授权和Label机制，可以提供精确到列级别的安全方案。
- AnalyticDB支持同时新建纯ssd存储实例、ssd与sata混合存储实例，两种实例可以在同一集群内同时存在，极大得提高了用户对集群的使用效率。
- AnalyticDB支持离线导入数据和实时写入数据，实时写入生产可达到200万行每秒。
- DataWorks支持多人协同开发，支持多种任务调度模型，支持统一数据及权限管理。

5 典型场景

在使用大数据轻量专有云过程中，大数据轻量专有云各产品会形成数据链路闭环。

大数据轻量专有云各产品形成的数据链路闭环，如下图所示。



将源数据写入到AnalyticDB的方式如下：

- 支持从MaxCompute同步数据到AnalyticDB：
  1. 用户离线数据通过数据开发套件DataWorks提供的数据集成功能，从源数据库中将数据同步到MaxCompute的project中；用户实时数据通过实时写入程序实时写入到MaxCompute的project中。
  2. 用户根据业务逻辑在数据开发套件中开发任务，可以是ETL类任务、与业务逻辑相关的任务、多个有依赖性的任务。
  3. 用户开发同步任务将数据批量离线同步到AnalyticDB。
- 支持通过实时写入程序将实时数据以标准INSERT语句写入到AnalyticDB。

关系网络分析I+、大数据应用加速器DTBoost及合作伙伴都对接到分析型数据库，由分析型数据库提供高并发、低时延的多维查询功能。

AnalyticDB支持创建高性能和大存储两种数据库实例。高性能实例仅使用SSD盘作为数据存储；大存储实例使用SSD盘与SATA盘混合存储。因此，大存储实例单GB的成本更低。

6 MaxCompute

6.1 产品概述

大数据计算服务（MaxCompute）是基于飞天分布式平台，由阿里云自主研发的海量数据离线处理服务。MaxCompute提供针对TB/PB级别数据、实时性要求不高的批量处理能力，主要应用于日志分析、机器学习、数据仓库、数据挖掘、商业智能等领域。

6.2 产品特性和核心优势

产品特点

- MaxCompute是面向大数据处理的分布式系统，主要提供结构化数据的存储和计算，是阿里巴巴云计算整体解决方案中最核心的主力产品之一，是阿里巴巴大数据平台的基础计算平台。MaxCompute中的多租户、数据安全、水平扩展等特性是MaxCompute的核心设计目标，采用抽象的作业处理框架为不同用户对各种数据处理任务提供统一的编程接口和界面。
- 采用分布式架构，规模可以根据需要平行扩展。
- 自动存储容错机制，保障数据高可靠性。
- 所有计算在沙箱中运行，保障数据高安全性。
- 以RESTful API的方式提供服务。
- 支持高并发、高吞吐量的数据上传下载。
- 支持离线计算、机器学习两类模型及计算服务。
- 支持基于SQL、Mapreduce、Graph、MPI等多种编程模型的数据处理方式。
- 支持多租户，多个用户可以协同分析数据。
- 支持基于ACL和policy的用户权限管理，可以配置灵活的数据访问控制策略，防止数据越权访问。
- 支持Spark的应用，即Spark on MaxCompute。

产品优势

- **海量运算触手可得**：用户不必关心数据规模增长带来的存储困难、运算时间延长等烦恼，根据用户的数据规模自动扩展集群的存储和计算能力，使用户专心于数据分析和挖掘，最大化发挥数据的价值。
- **服务“开箱即用”**：用户不必关心集群的搭建、配置和运维工作，仅需简单的几步操作，用户便在MaxCompute中上传数据、分析数据并得到分析结果。

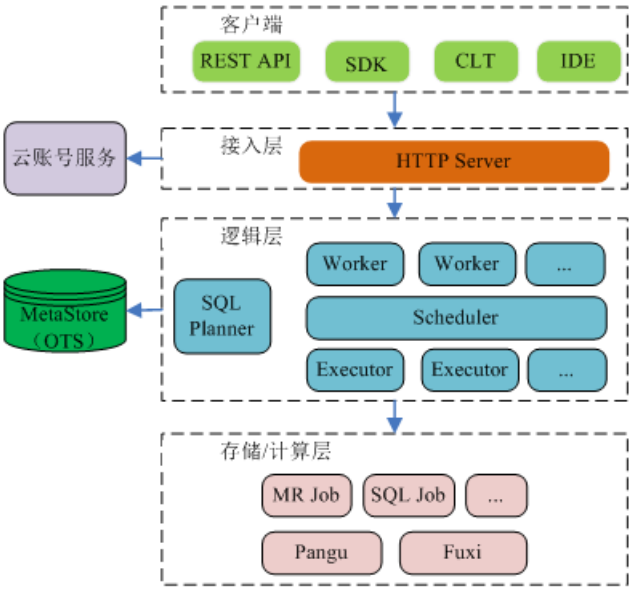
- **数据存储安全可靠**：采用多副本技术、读写请求鉴权、应用沙箱、系统沙箱等多层次数据存储和访问安全机制来保护用户的数据，使其不丢失、不泄露、不被窃取。
- **多用户协作**：通过配置不同的数据访问策略，用户可以让组织中的多名数据分析师协同工作，并且每人仅能访问自己权限许可内的数据，在保障数据安全的前提下最大化工作效率。

6.3 产品架构

6.3.1 功能架构

MaxCompute的功能架构如图 *6-1: MaxCompute*功能架构图所示：

图 6-1: MaxCompute功能架构图



MaxCompute由四部分组成，分别是**客户端**、**接入层**、**逻辑层**及**计算层**，每一层均可平行扩展。

MaxCompute的客户端有以下几种形式：

- **API**以RESTful API的方式提供离线数据处理服务。
- **SDK**：对RESTful API的封装，目前有Java等版本的实现。
- **CLT**（**Command Line Tool**）：运行在Window/Linux下的客户端工具，通过CLT可以提交命令完成Project管理、DDL、DML等操作。
- **DataWorks**：提供了上层可视化ETL/BI工具，用户可以基于DataWorks完成数据同步、任务调度、报表生成等常见操作。

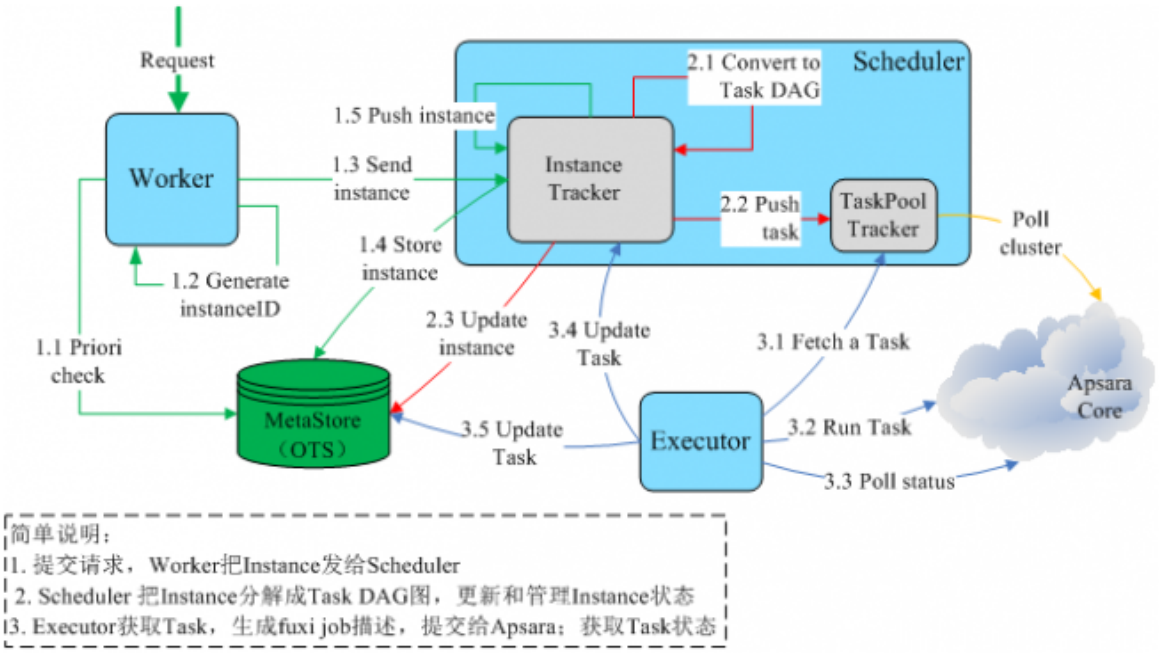
MaxCompute接入层提供HTTP（HTTPS）服务、Load Balance、用户认证和服务层面的访问控制。

MaxCompute逻辑层是核心部分，实现用户空间和对象的管理、命令的解析与执行逻辑、数据对象的访问控制与授权等功能。逻辑层包括两个集群：调度集群与计算集群。调度集群主要负责用户空间和对象的管理、Query和命令的解析与启动、数据对象的访问控制与授权等功能；计算集群主要负责task的执行。控制集群和计算集群均可根据规模平行扩展。在调度集群中有Worker、Scheduler和Executor三个角色，其中：

- **Worker处理所有RESTful请求**：包括用户空间（project）管理操作、资源（resource）管理操作、作业管理等，对于SQL、MapReduce、Graph等启动Fuxi任务的作业，会提交Scheduler进一步处理。
- **Scheduler负责instance的调度**：包括将instance分解为task、对等待提交的task进行排序、以及向计算集群的Fuxi master询问资源占用情况以进行流控（Fuxi slot满的时候，停止响应Executor的task申请）。
- **Executor负责启动SQL/ MR task**：向计算集群的Fuxi master提交Fuxi任务，并监控这些任务的运行。

简单的说，当用户提交一个作业请求时，接入层的Web服务器查询获取已注册的Worker的IP地址，并随机选择选择某些Worker发送API请求。Worker将请求发送给Scheduler，由其负责调度和流控。Executor会主动轮询Scheduler的队列，若资源满足条件，则开始执行任务，并将任务执行状态反馈给Scheduler。其对应的业务执行逻辑图如下图所示：

图 6-2: 业务执行逻辑图



MaxCompute存储与计算层为阿里云自主知识产权的云计算平台的核心构件，是飞天内核，运行在和控制集群独立的计算集群上。上面的MaxCompute架构图中仅列出了若干飞天内核主要模块，例如：盘古（Pangu）、伏羲（Fuxi）等。

其中，**盘古（Pangu）**是一个分布式文件系统，其设计目标是将大量通用机器的存储资源聚合在一起，为用户提供大规模和可靠的分布式存储服务，是飞天内核的重要组成部分。

盘古目前包括三台master和多台chunkserver，前者负责文件meta信息的存储和管理，后者负责数据存储。为了保证可靠性，同一个数据块会被存储在多个chunkserver上，一般情况下盘古保存的数据会有3份副本，所有的MaxCompute数据文件都保存在盘古上，在 /product/aliyun/odps盘古目录下可以找到。需要注意，其中master是热备，同一时间只有一台在工作。

- **master**：
  1. 盘古master维护了整个文件系统的元数据，其中包括命名空间，文件到数据块的映射及数据块的存储地址等。
  2. 作为整个分布式文件存储系统的大脑，盘古master控制着系统层面的活动如孤立数据块的垃圾回收、chunckserver间的数据合并、判断chunkserver是否健康、恢复由于chunkserver宕机所造成的数据块丢失等。
  3. 盘古master同时还负责调节同一时间片中多台客户端对数据的访问来维护同一集群中数据的完整性。

4. 盘古master只对客户端提供元数据相关的操作，而数据传输相关的通讯都直接在chunkservers间进行。

- **chunkserver**：盘古上的文件会被切成多个固定大小的存储单元，每个存储单元都叫一个chunk。存储数据块chunk的机器称为chunkserver，对每个数据块，创建之初，pangumaster都会分配一个128位的ID，盘古客户端根据ID来读取存储在磁盘上的数据块。

同时，为了更好地支持MaxCompute表中的结构化数据，MaxCompute使用统一的数据文件格式，实现了一种特殊格式的Pangu文件，称为CFILE。

- CFile是一种基于列存储的文件格式，其主要目的是为了降低离线数据处理过程中的无效磁盘读取操作。文件中的数据以列为单位聚簇组织（聚簇被称为Block），并在存储到文件系统之前进行压缩，减少了存储空间的占用。在离线数据处理的场景中，用户只需要读取待处理的数据，避免了无用的磁盘操作，提高读取的磁盘效率，同时减少了网络带宽的使用。
- CFile文件的存储结构逻辑上可以分为三个区域，分别为数据区（Data区）、索引区（Index区）以及元信息区（Header区）。其中，数据区存储的是按照列划分，以Block为组织单位的用户数据；索引区存储的是每列的数据Block对应的索引，其中包含每个Block在文件中的起始位置、Block压缩后的长度以及Block内的数据个数（对非定长的数据类型如string而言）。元信息区存储了该文件中每一列的元信息，例如该列的索引在文件中的起始位置以及索引长度、该列的类型信息、压缩方法等，以及文件中用户数据的行数以及版本号等。
- 目前支持如下五种原始的数据类型（即MaxCompute支持该五种类型）：
  - BIGINT，8字节有符号整型。
  - BOOLEAN，布尔型，包括TRUE/FALSE。
  - DOUBLE，8字节双精度浮点数。
  - STRING，字符串，需要注意在MaxCompute中的函数会假设STRING中存的是UTF8编码的字符串，其他编码格式可能会导致异常。
  - DATETIME日期类型，格式为**YYYY-MM-DD HH:mm:SS**，例如：2012-01-02 10:09:25。

而**伏羲**（**Fuxi**）则是飞天平台内核中负责资源管理和任务调度的模块，也为应用开发提供了一套编程基础框架。其主要功能是充分利用整个集群的硬件资源来服务用户和系统的计算需求。伏羲同时支持两种应用类型的计算：低延迟的在线服务和高吞吐的离线处理，分别称为Fuxi Service和Fuxi Job，可以对应Hadoop中的YARN。

- **Fuxi Service**：Service是伏羲上的常驻进程，由用户申请创建及销毁，伏羲不会主动销毁Service进程。

- **Fuxi Job**：Job是伏羲上的临时任务，一旦任务结束，资源就会被释放，由伏羲回收。

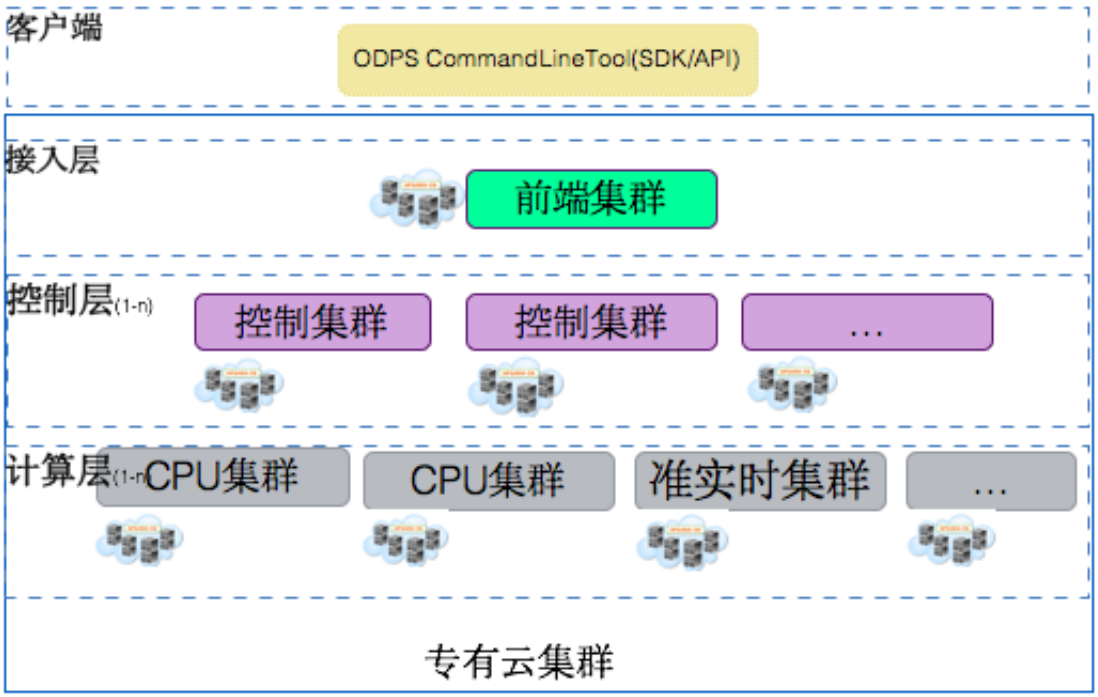
在资源管理上，伏羲负责调度和分配集群的存储、计算等资源给上层应用，支持计算资源额度、访问控制和作业优先级，保证有效的资源共享。在任务调度上，伏羲提供了数据驱动的多级流水线并行计算框架，类似于MapReduce编程模式，适用于海量数据处理和大规模计算等复杂应用。

伏羲目前包括两台master和多台tubo，其中master是冷备，同时只有一台在工作。而每台计算节点上都启动了一个tubo进程，用来管理单机资源，例如汇总整机的可用的CPU、内存、硬盘、网络，同时记录每台机器已被占用的资源，每个机器上的tubo进程会将这些资源汇报给伏羲master，由伏羲master统一管理及调度。

6.3.2 系统架构

MaxCompute的系统架构如图 6-3: MaxCompute系统架构图所示：

图 6-3: MaxCompute系统架构图



6.4 产品价值

MaxCompute与传统数据库相比，具有如下的价值点：

表 6-1: 价值点对比

价值点	传统数据库	MaxCompute
系统扩展能力	共享磁盘技术，难以超过100节点，分库分表技术将导致应用数据碰撞、比对等大规模计算带来海量数据的数据交换开销，极大限制应用的分析能力。	超过10000节点，支持超过1.5EB数据量。例如阿里双11活动中，6小时内MaxCompute处理数据量超过300PB。
数据类型支持	不适用于非结构化计算。	同时支持结构化、非结构化数据处理。
高可用性	无冗余存储，传统备份恢复对PB级数据量不可用，单点磁盘故障会导致全库不可用。	无共享的多副本技术，无单点故障。
复杂计算能力	不支持迭代计算，不支持图计算。共享磁盘技术，复杂计算导致节点间大量数据交换，带宽难以支持。	分布式存储，支持MR、SQL、迭代计算、MPI、图计算等多种计算框架。
并发能力	一个大规模计算任务可能会消耗掉全部系统资源（如索引计算），存在网络、热点盘（数据字典）等多种瓶颈，无法支持高并发访问。	具有完善的多租户资源隔离和资源管理工具，用户对集群资源一目了然，并且很方便的管理每个业务使用的资源量，可以支持超过1万的并发访问。
性能支持	索引机制导致实时入库数据难以支持分析型应用，海量数据碰撞比对、分析预测计算时间可能超过24小时，性能无法满足需求。	关注与海量数据的并行计算能力，支持数据实时入库可用、具备高性能大规模离线计算、海量数据实时多维分析、流计算等多种高性能计算能力。

6.5 功能描述

6.5.1 Tunnel

Tunnel是MaxCompute提供的数据通道服务，各种异构数据源都可通过Tunnel服务导入MaxCompute或从MaxCompute导出。它是MaxCompute数据对外的统一通道，提供高吞吐、持续稳定的服务。

Tunnel提供了Restful API接口，提供了Java SDK，可以方便用户编程。

6.5.2 SQL

MaxCompute SQL适用于海量数据（TB级别），实时性要求不高的场合，它的每个作业的准备、提交等阶段要花费较长时间，因此要求每秒处理几千至数万笔事务的业务是不能用MaxCompute SQL完成的。

MaxCompute SQL是一种结构化查询语言，语法和Oracle/MySQL/Hive SQL类似，可以看作是标准SQL的子集，但不能因此简单的把MaxCompute SQL等价成一个数据库，它在很多方面并不具备数据库的特征，如事务、主键约束、索引等。

6.5.3 MapReduce

MapReduce是一种编程模型，基本等同Hadoop中的MapReduce。用于大规模数据集（TB级别）的并行运算MaxCompute。

用户可以使用MapReduce提供的接口（Java API）编写MapReduce程序处理MaxCompute中的数据。概念“Map（映射）”和“Reduce（归约）”和它们的主要思想，都是从函数式编程语言和矢量编程语言中借鉴来的特性。它极大地方便了编程人员在不会分布式并行编程的情况下，将自己的程序运行在分布式系统上。

当前的软件实现是指定一个Map（映射）函数，用来把一组键值对映射成一组新的键值对，指定并发的Reduce（归约）函数，用来保证所有映射的键值对中的每一个共享相同的键组。

MaxCompute MapReduce特性：

- Hadoop-style，针对MaxCompute场景设计（用于处理Table和Volume）。
- 输入输出仅支持MaxCompute内置类型。
- 可以输入多表，输出多表或到不同分区。
- 可以读资源（Resource）。
- 不支持输入view。
- 受限的沙箱安全环境。

6.5.4 Graph

Graph是MaxCompute提供的面向迭代的图计算处理框架，为用户提供类似Pregel的编程接口，用户可以基于Graph框架开发高效的机器学习或数据挖掘算法。

在互联网环境下，存在很多海量图结构的数据，例如社交网络、物流信息等，这类图计算模型的典型特点是迭代，整个计算过程是通过一轮一轮反复迭代求解，最后达到一个收敛状态。例如对于

需要迭代学习模型参数的机器学习算法而言，图计算模型比MapReduce有天然优势。在实际应用中，用户将问题抽象成图，然后以顶点为中心，通过超步进行迭代更新。

MaxCompute Graph目前提供两种模式：

- 离线模式：适用于计算规模较大的场景，类似于MapReduce作业，每次运行完成加载和计算两个过程。
- 交互模式：适用于计算规模较小的场景，用户实现UDF，然后通过命令行方式交互。

在离线模式下，加载和计算是两个独立的步骤，数据加载后会常驻内存，用户可以对数据执行不同的计算逻辑。例如风控部门每天会加载一次数据，运营人员会对这份数据执行不同的查询逻辑，查看数据之间的关系。

在阿里巴巴内部，MaxCompute Graph已经有很多应用，例如实现带权重的PageRank算法计算支付宝用户身边影响力指数；实现变分贝叶斯EM模型，基于用户购买的商品属性信息，推测用户的汽车品牌分布等。

6.5.5 访问并处理非结构化数据

MaxCompute团队依托MaxCompute系统架构，引入非结构化数据处理框架，解决MaxCompute SQL面对MaxCompute表外的各种用户数据时（例如：OSS上的非结构化数据或者来自TableStore（OTS）的非结构化数据），需要首先通过各种工具导入MaxCompute表，才能在其上面进行计算，无法直接处理的问题。

用户只需要通过一条简单的DDL语句，在MaxCompute上创建一张外部表，建立MaxCompute表与外部数据源的关联，即可以为各种数据在MaxCompute上的计算处理提供入口。并且创建好的外部表可以像普通的MaxCompute表一样使用，充分利用MaxCompute SQL的强大计算功能。

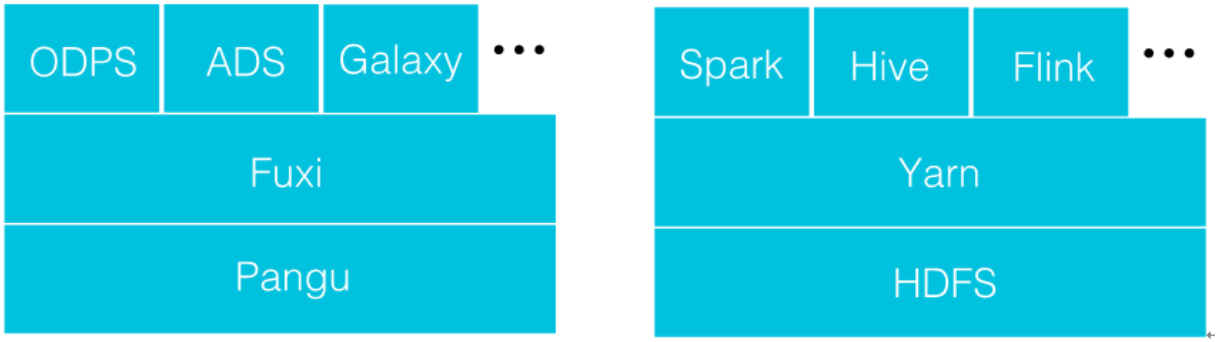
6.5.6 Spark on MaxCompute

6.5.6.1 开源平台——Cupid

MaxCompute是阿里云自主研发的大数据解决方案，其规模和稳定性在业界都是领先的。大数据开源社区当前也非常活跃，应对各类需求的系统快速出现并发展。为了更好服务用户，同时也为了丰富MaxCompute的生态，MaxCompute团队研发了Cupid平台，把MaxCompute跟开源社区连接到一起，兼顾开源社区的多样性和飞天系统的规模和稳定性。

开源和飞天系统的软件栈比较相似，如下图所示。

图 6-4: 软件栈对比

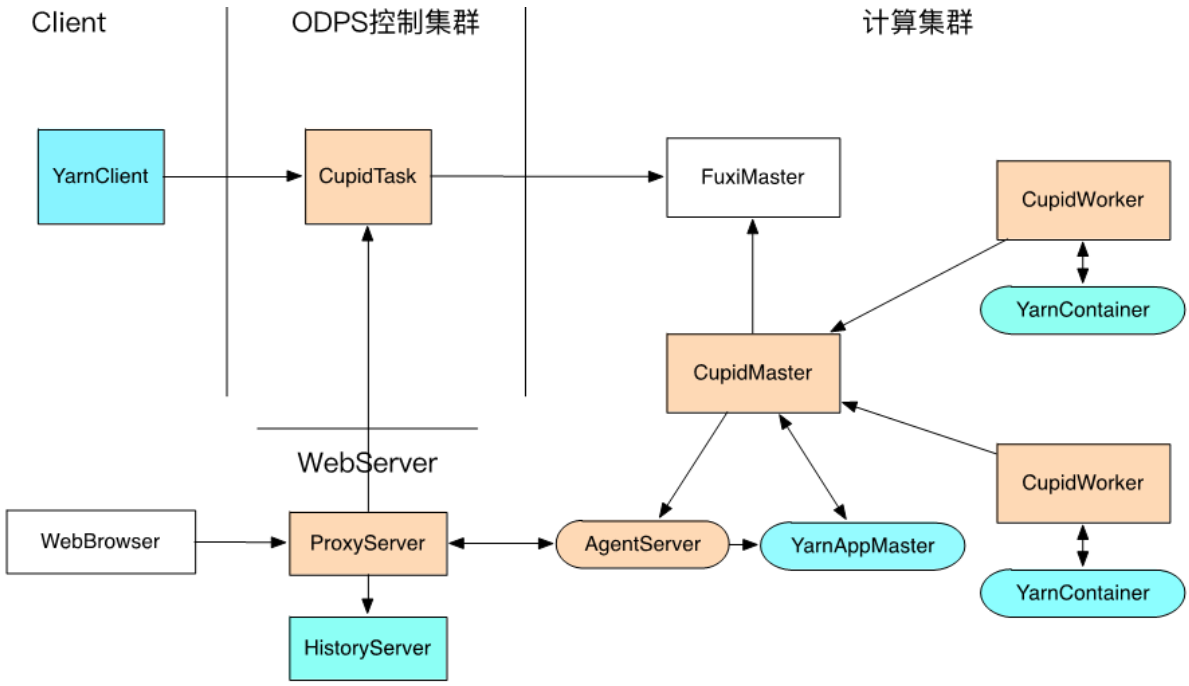


对于分布式文件系统，社区大多使用HDFS，飞天里面则是使用Pangu；对于分布式调度系统，社区用的比较多的是Yarn，飞天系统里面则是使用Fuxi；在此之上是应对各类场景的计算引擎。为了能够把开源应用（例如Spark）运行在MaxCompute系统中，Cupid首先要做兼容。

6.5.6.1.1 兼容Yarn

Yarn面向应用的接口主要有三个：YarnClient、AMRMClient和NMClient。YarnClient用于客户在应用端提交作业到集群；AMRMClient用于AppMaster向ResourceManager申请和释放资源；NMClient用于启动和停止应用的Container。

图 6-5: Yarn on MaxCompute



一个Yarn上App执行到MaxCompute平台的流程如上图所示，其中黄色部分是Cupid平台的组件，浅蓝色部分是开源组件。具体流程如下：

1. 用户通过YarnClient访问MaxCompute控制集群，提交作业给FuxiMaster。
2. FuxiMaster拉起来一个CupidMaster，CupidMaster按照Yarn的协议把YarnAppMaster拉起来。
3. YarnAppMaster通过CupidMaster跟FuxiMaster交互，申请和释放资源。
4. 启动新的Container时，fuxi的tubo会先启动一个CupidWorker，CupidWork根据Yarn的协议拉起Container。

说明：
YarnAppMaster中一般都会有UI，Cupid通过代理机制实现。

6.5.6.1.2 兼容FileSystem

社区里面大多使用HDFS作为分布式存储，Hadoop社区提供了FileSystem接口，目前阿里云的oss，亚马逊的s3都兼容了这个接口。MaxCompute团队同样实现了Pangu兼容FileSystem接口，提交到MaxCompute集群的开源job同样可以原封不动运行起来，并且获得原生pangu的性能。

说明：
需要注意，Pangu不直接对外提供服务，Pangu上面的数据只能作为作业中间数据，例如checkpoint之类的。如果用户想要使存入的数据在其他环境同样能够访问，目前可以选择使用OSS。

6.5.6.1.3 DiskDrive

开源应用大多会使用本地文件系统来进行数据处理，例如spark shuffle、storage等。在大集群环境下，磁盘是一种非常重要的系统资源，应该被统一管理，才能保证高可用、高性能以及安全性。在飞天系统中，磁盘都是被pangu统一管理。为了在pangu的基础上提供本地文件系统接口，Cupid把网盘引入到MaxCompute，设计并实现了DiskDriverService系统。

6.5.6.2 功能扩展

由于MaxCompute提供了Cupid框架来支持开源应用，因此，Spark可以在MaxCompute上面运行起来。为了更好的方便用户使用，也为了更好地跟MaxCompute融合，Spark on MaxCompute需要对Spark进行功能扩展，主要包括：保证Spark开源程序的安全隔离；跟MaxCompute数据打通；在多租户的集群中提供交互式。

6.5.6.2.1 安全隔离

Spark作业提交到MaxCompute计算集群中，为防止用户恶意代码对系统进行攻击，Spark作业需要运行在安全沙箱里面。整体使用父子进程架构，Cupid框架代码运行在父进程中，Spark代码运行在子进程中。Spark需要访问系统服务时（例如Fuxi、Pangu），可以通过父子进程通信的方式，由父进程代理访问。

6.5.6.2.2 数据互通

运行在MaxCompute中的Spark的一个优势是，Spark作业和MaxCompute作业在整个集群资源上是统一的，可以做到相互之间数据直接访问，而不需要跨级群拖数据。

数据互通包括两个方面：元数据和存储数据，并且Spark存储MaxCompute数据必须通过MaxCompute账号体系进行鉴权。**Spark on MaxCompute**实现了OdpsRDD、OdpsDataFrame，可以满足用户对Spark这两种API的需求。同时Spark SQL可以直接访问MaxCompute元数据进行SQL优化，并且在物理层直接存取MaxCompute数据格式。

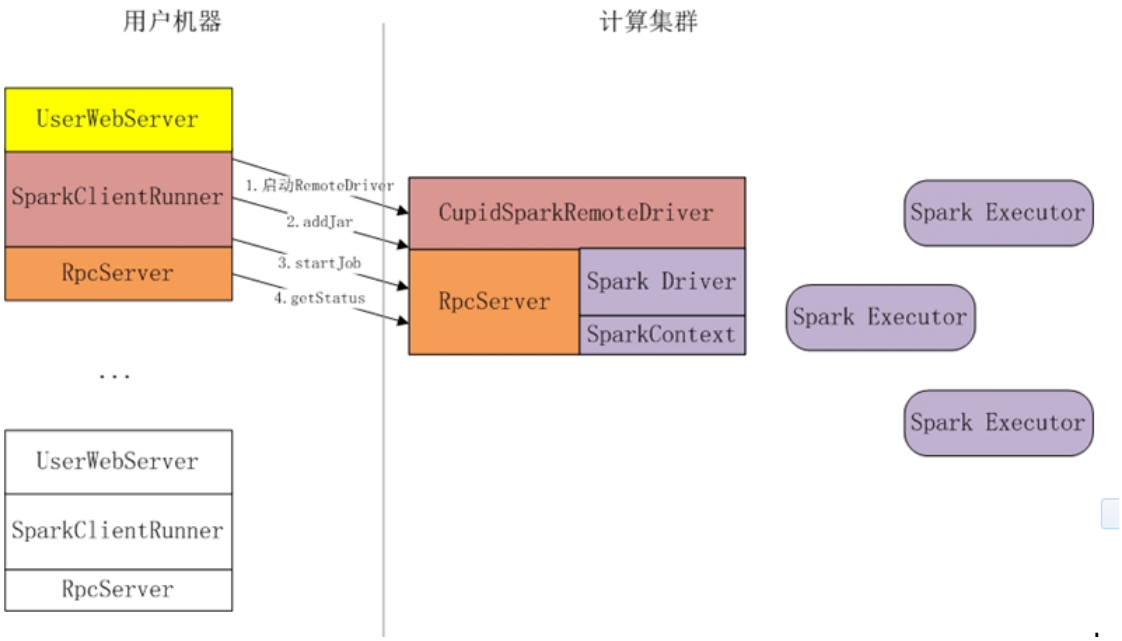
6.5.6.2.3 Client模式

社区spark生产主要使用**yarn-cluster**、**yarn-client**两种模式。**yarn-cluster**模式将spark作业提交到集群运行，运行完毕后客户端打印状态日志。这种模式无法向一个Spark Session动态多次提交作业，且客户端无法获取每个job的状态及结果；而**yarn-client**模式，主要解决spark交互式场景问题，需要在客户端机器启动Driver，无法将Spark Session作为一个服务。因此MaxCompute团队基于**Spark on MaxCompute**开发了**Client**模式来解决上述的问题。该模式具有以下特点：

1. 客户端轻量级，不用再启动spark的Driver。
2. 客户端有一套API向MaxCompute集群的同一个Spark Session动态提交作业，并监控状态。
3. 客户端可以通过监控作业状态及结果，构建作业之间的依赖关系。
4. 用户可以动态编译应用程序jar，通过客户端提交到原有的Spark Session运行。
5. 客户端可以集成在业务的WebServer中，并且可以进行水平扩展。

Client模式会先通过CupidSparkClientRunner在MaxCompute集群启动一个Spark Session。后续通过CupidSparkClientRunner在客户端提交作业、获取作业状态及运行结果，同时各个作业之间可以共享cache的数据等；也可以构建多个CupidSparkClientRunner与同一个Spark Session交互，运行时的框架图如下图所示。

图 6-6: Spark Client模式



具体使用Spark Client模式的执行流程如下所示：

1. 向MaxCompute集群提交启动CupidSparkRemoteDriver，并获取SparkClientRunner对象。
2. 通过SparkClientRunner添加要执行作业的jar包到RemoteDriver。
3. SparkClientRunner使用job的classname向RemoteDriver提交运行。
4. SparkClientRunner通过提交作业返回的job id来监控作业的状态。

6.5.6.2.4 Spark生态支持

Spark拥有丰富的生态：Mllib、Streaming、PySpark、SparkR、Graphx、SQL。**Spark on MaxCompute**对Spark的完整生态提供支持，使用方式跟社区保持一致。此外，**Spark on MaxCompute**也支持Spark UI以及历史日志的访问。

6.6 应用场景

MaxCompute主要面向三类大数据处理场景：

- 基于SQL构建大规模数据仓库和企业BI系统。
- 基于MapReduce和MPI的分布式编程模型开发大数据应用。
- 基于统计和机器学习算法，开发大数据统计模型和数据挖掘。

6.6.1 搭建数据仓库

图 6-7: 搭建数仓

使用MaxCompute可以轻松打造一个云端数据仓库，借助MaxCompute的分区、数据表统计、表的生命周期等功能，用户可以很方便的实现数据仓库的历史信息增强存储、冷热表区分、数据质量控制等场景。

阿里金融数仓团队基于MaxCompute构建了一个完善复杂、功能强大的数据仓库体系，包含六个层次：源数据层、ODS层、企业数据仓库层、通用维度模型层、应用集市层和展现层。

- 源数据层处理各个来源数据，包括淘宝、支付宝、B2B、外部数据等。
- ODS作为数据导入的临时存储层。
- 企业数据仓库层采用3NF建模方式，按主题（如商品、店铺）进行划分，包括完整的历史数据。
- 通用维度模型以维度建模方式构建面向通用业务应用的模型层，不以满足特定的应用为目的。通用维度模型层的目的是屏蔽业务需求变化，以一致性维度和事实的方式为上层提供数据。
- 应用集市层是面向需求，构建满足某一应用需求的数据集市。
- 展现层提供一些数据门户（Portal）和服务等，供应用访问。

在这个体系架构中，不可避免会涉及元数据管理等其他方面。

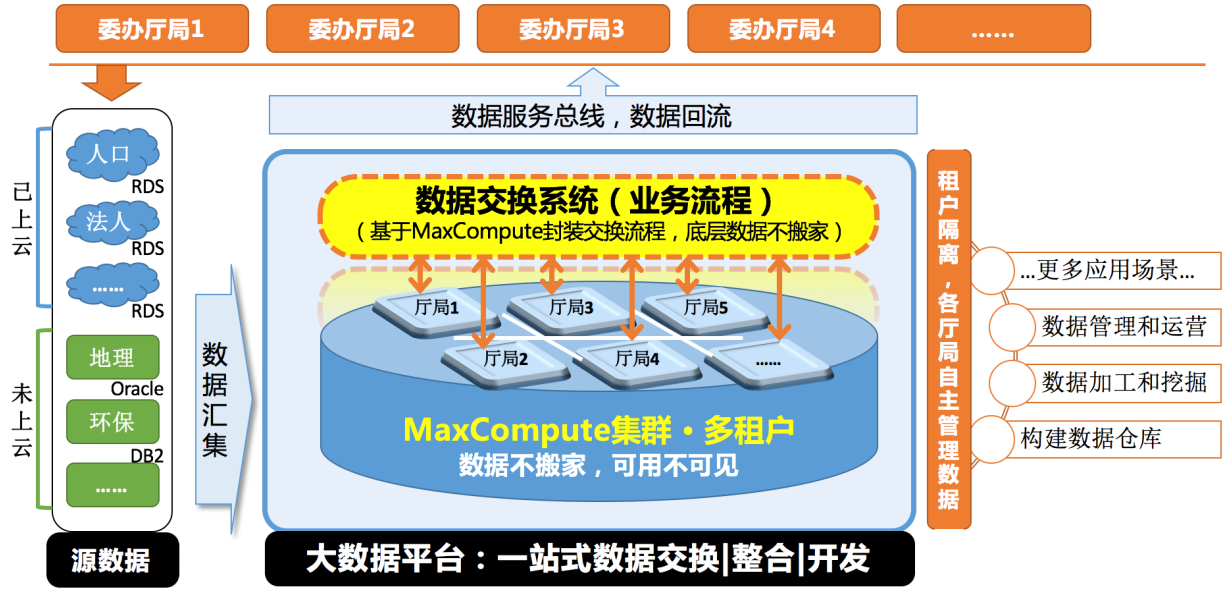
金融的数据仓库主要是基于MaxCompute SQL完成离线计算，并通过一系列指标规则和算法完成离线决策，输出结果给在线决策使用。

跟传统数据库相比，基于MaxCompute搭建数仓，有以下不同之处：

- **历史数据存储**：MaxCompute天生支持大数据，不必像传统数据库那样将历史数据转储到廉价存储媒介。
- **分区方式**：传统数据库支持的分区方式更丰富一些（例如支持范围分区），但在数仓场景下，MaxCompute目前支持的分区方式基本够用。不管用哪个数据库搭建数仓，表分区的设计理念和原则一致。
- **大宽表**：MaxCompute按字段存储，建大宽表有天然优势。
- **数据整合**：传统数据库都用存储过程来加工、整合数据，MaxCompute则需要将逻辑拆分成一段段SQL，虽然实现途径不同，但算法是一致的。经过几年的使用比较，采用分段SQL的方式更清晰和高效，而存储过程的方式更灵活且具备处理复杂逻辑的能力。

6.6.2 大数据共享及交换

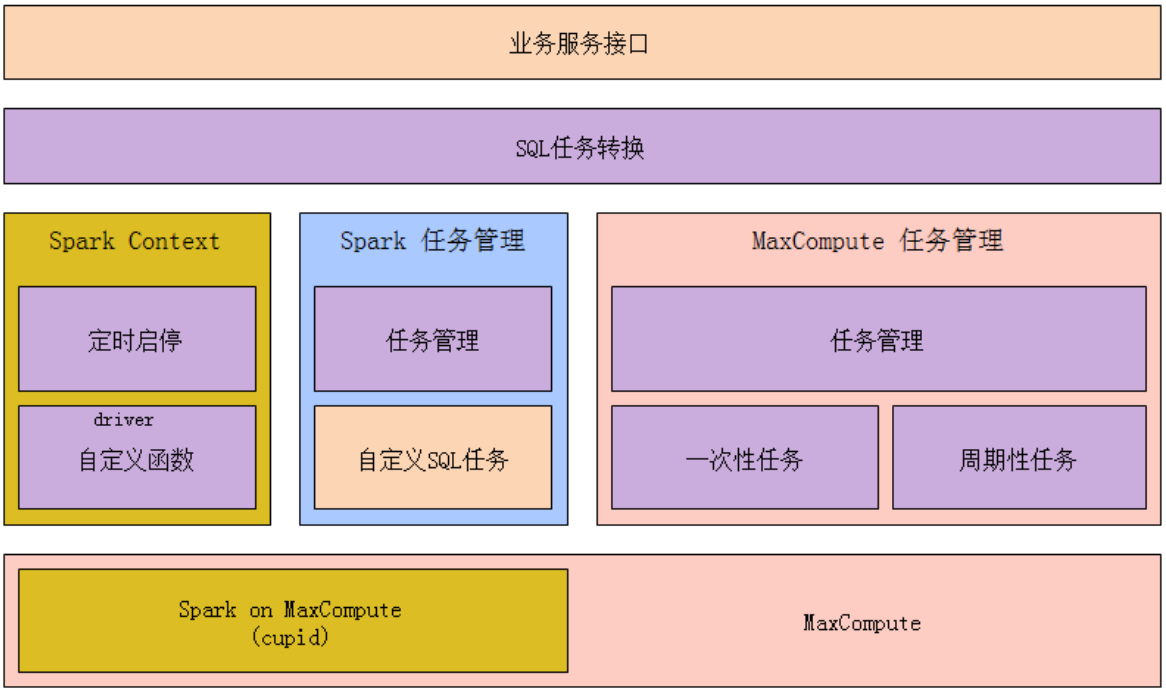
图 6-8: 大数据共享及交换



MaxCompute具有丰富的多种权限管理方式、灵活数据访问控制策略。MaxCompute提供丰富的授权管理手段，包括ACL、角色授权、 Policy授权、跨Project授权以及Label机制，可以提供精确到列级别的安全方案，满足一个组织或者跨组织间的授权需求。安全要求较高的项目，可以提供项目保护机制，防止数据泄露，而且对用户的任何操作都记录日志便于事后追溯审计。

6.6.3 Spark on MaxCompute典型实践

图 6-9: 典型实践



Spark on MaxCompute支持Client Mode的业务计算平台应用，应用框架如上图所示。

6.7 性能与可靠性

做为海量处理数据平台，MaxCompute单一集群规模可以达到10000+台服务器，而且支持多集群技术（用户访问方式不变，MaxCompute各个角色均可平行扩展）。

计算性能方面：除MaxCompute采用C++研发本身语言性能优势外，还在引擎层面引入基于代价优化和运行时LLVM及向量化处理等技术和手段，使得计算效率上优于同类型的开源产品。

高可用性方面：MaxCompute采用无单点架构和多副本机制，同时支持Failover自动副本和服务节点无缝切换（资源充足前提下）；支持宕机服务节点自动迁移（资源充足前提下），宕机计算节点自动迁移及宕机部分管理节点自动迁移，不影响大数据计算整体服务。

数据可靠性方面：MaxCompute是通过多副本技术实现的，数据存储可达到三副本可靠性。MaxCompute底层采用分布式文件系统，通过将数据副本分布到不同时效的多台硬件上，能够保证正常概率损坏的硬件下的数据安全，单节点故障或磁盘单点故障时均不影响作业的正常运行。

具体的性能指标如下表所示：

表 6-2: 性能指标

规格项	规格描述	性能	扩展性	备注
MapReduce性能指标	Wordcount：字符计数速率（GB／分钟）	8.5GB／分钟	水平扩充。如10台则85GB／分钟	-
	Terasort：对输入文件按Key进行全局排序速率（GB／分钟）	13.55GB／分钟	水平扩充。如10台则135.5GB／分钟	-
SQL性能指标	SQL Join：执行odps sql join操作的速率（GB／分钟）	31.44GB／分钟	水平扩充。如10台则314.4GB／分钟	-
	SQL Aggregation：执行odps sql Aggregation操作的速率（GB／分钟）	64GB／分钟	水平扩充。如10台则640GB／分钟	-
Tunnel性能指标	Upload：通过tunnel命令上传数据到MaxCompute的速率（GB／分钟）	15MB／秒	水平扩充。如10台则150MB／秒	在外部有dns服务器情况下标准规格与单机是倍数关系，如果是多线程，可将达到服务器网卡限制速率，千兆网卡，可达到100MB/秒
	Download：通过tunnel命令从MaxCompute下载数据到本地的速率（GB／分钟）	30MB／秒	水平扩充。如10台则300MB／秒	在外部有dns服务器情况下标准规格与单机是倍数关系，如果是多线程，可将达到服务器网卡限制速率，千兆网卡，可达到100MB/秒
硬件性能指标	CPU：cpu数	64核	水平扩充	通过grep proc /proc/cpuinfo获得

规格项	规格描述	性能	扩展性	备注
	Memory：内存数	192GB	水平扩充	考虑了系统消耗和换算差异，通过free -g获得
	IOPS：每秒进行读写次数（Input/Output Per Second）	参考服务器数据	水平扩充	IOPS数据与实际业务相关，建议使用服务器的硬盘读写数据
	容量：可存储数据量（TB）	6T*11	水平扩充	考虑了系统消耗和换算差异，通过df - h获得

7 大数据开发套件

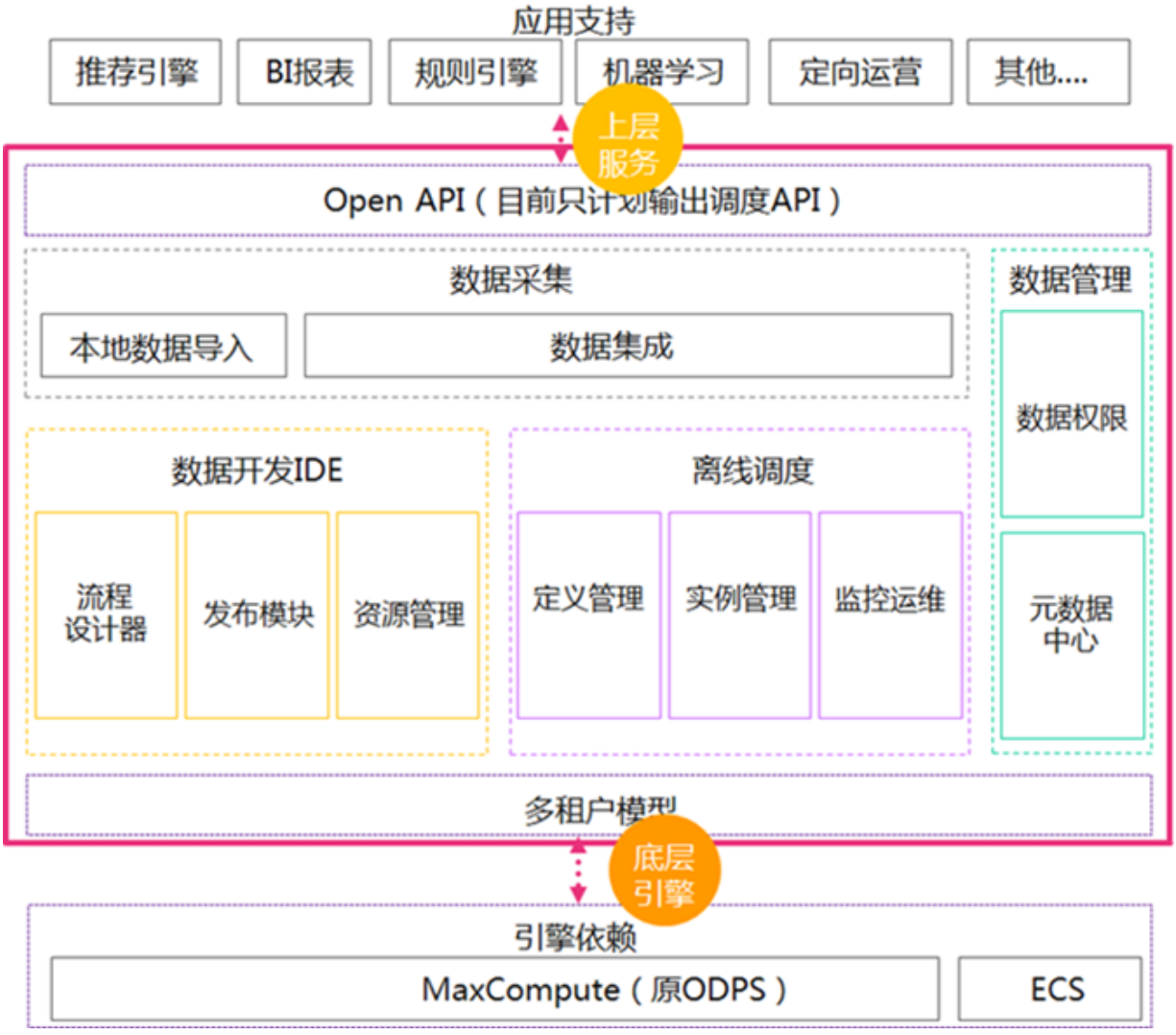
7.1 产品概述

DataWorks是阿里云推出的一款大数据领域平台级产品。面向企业和个人用户提供端到端的一站式大数据开发、管理、离线调度和应用解决方案。

DataWorks以 **enable更多人使用更多数据** 为使命，提供DT时代的大数据基础服务：

- enable大型企业构建PB、EB级别的数据仓库，实现超大规模数据集成，对数据进行资产化管理，通过对数据价值的深度挖掘实现业务的数据化运营。
- enable中小企业和个人用户快速构建数据应用，助力中小企业的数据业务创新。

图 7-1: 产品组成



DataWorks产品由数据开发IDE、离线调度系统、数据集成工具和数据管理四大部分组成：

- **数据开发IDE**：提供开箱即用的一站式数据开发工具，满足在线SQL、MR、Shell的编码工作，并提供多人协同开发和代码版本管理功能。通过可视化的流程设计工具可以满足快速构建数仓调度的依赖关系。
- **离线调度系统**：提供百万级的离线任务调度能力，以及在线运维、在线日志查询、调度状态监控报警等一系列功能。
- **数据集成工具**：提供海量异构数据源的数据集成能力，打通阿里云80%的数据库及存储设备的数据链路，以及常用关系型数据库、FTP、HDFS等多种数据链路，并且提供周期性定时集成的能力。
- **数据管理系统**，提供对MaxCompute（原ODPS）中以公司为单位的全量数据的管理能力，并提供权限管理、数据血缘和元数据查看等功能。

7.2 产品特性和核心优势

超大规模数据处理能力

DataWorks与阿里云大数据服务MaxCompute（原ODPS）天然集成，单个集群的规模可达5000台，并且具备跨机房的线性扩展能力，轻松处理海量数据。离线调度支持百万级任务量，实时监控告警。

核心指标：

- 万亿级数据JOIN，百万级并发job，作业I/O可达PB级/天。
- 具备跨集群（机房）数据共享能力，支持万级别的集群数，扩容不受限制。
- 提供功能强大易用的SQL、MR引擎，兼容大部分标准SQL语法。
- MaxCompute（原ODPS）采用三重备份、读写请求鉴权、应用沙箱、系统沙箱等多层次数据存储和访问安全机制来保护用户的数据，使其不丢失、不泄露、不被窃取。

一站式的数据开发环境

数据开发、离线调度、调度运维、监控告警、数据管理全流程串通。

核心指标：

- 一个产品，提供全流程所有功能。
- 提供可视化工作流程设计器功能，类似Kettle的工具，支持用户对流程进行设计并编辑。
- 多人协同作业机制，分角色进行任务开发、线上调度、运维、数据权限管理等功能，数据及任务无需落地即可完成复杂的操作流程。

海量异构数据源快速集成能力

提供11种异构数据源的数据读取能力，12种异构数据源的数据写入能力。并且提供脏数据过滤，流量控制等功能。

核心指标：

- 提供mysql、oracle、sqlserver、postgresql、rds、drds、MaxCompute、ftp、oss、hdfs、dm、sysbase的数据读取能力。
- 提供mysql、oracle、sqlserver、postgresql、rds、drds、MaxCompute、ads、ocs、oss、hdfs、dm、sysbase的数据写入能力。
- 提供脏数据过滤、流量控制能功能。
- 可以周期性调度，周期性数据集成能力。

Web化的软件服务

可在互联网/内部网络环境下直接使用，无需安装部署，拎包入驻，开箱即用。

多租户权限模型

多租户模型确保用户的数据被安全隔离，以租户为单位进行统一的权限管控、数据管理、调度资源管理和成员管理工作。

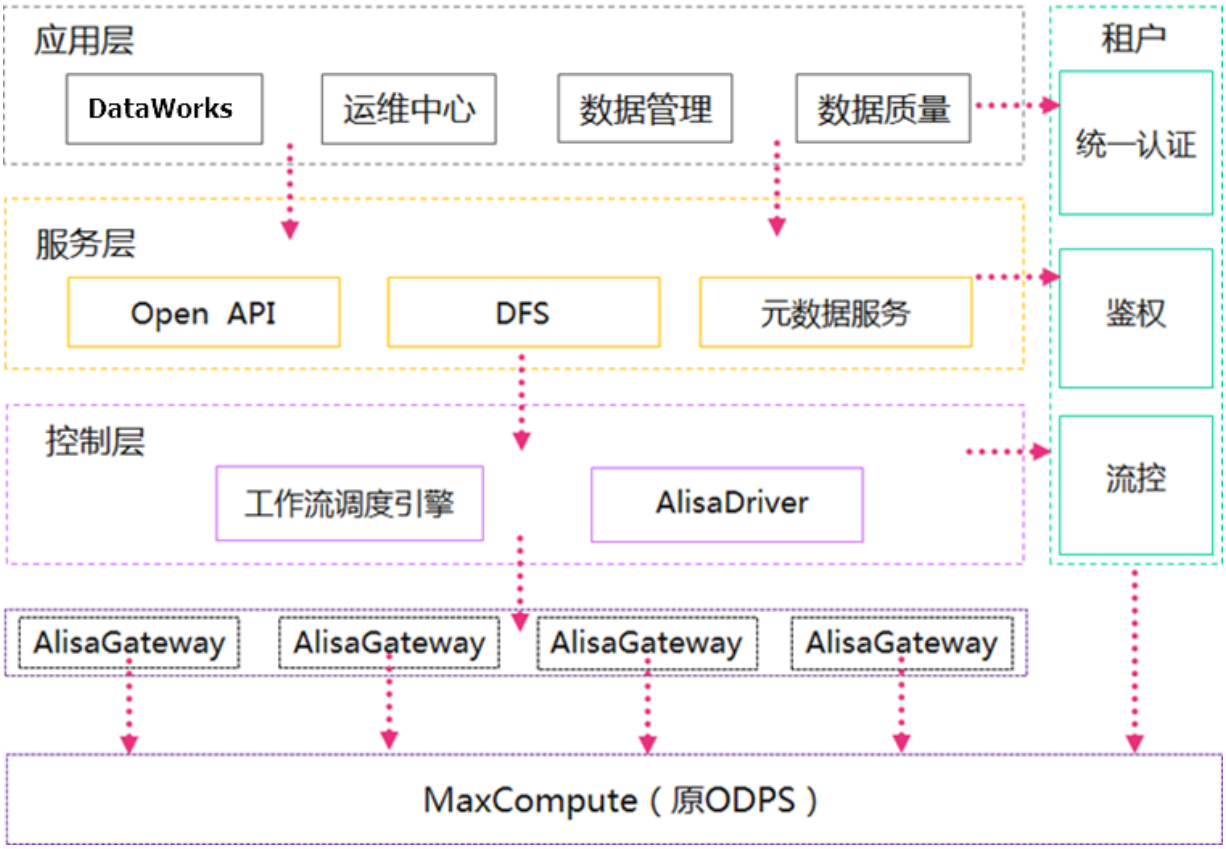
开放的平台

所有模块已实现组件化、服务化，用户可基于DataWorks的Open API来定制开发扩展功能。

7.3 系统架构

系统开放性架构

图 7-2: 系统架构图



DataWorks采用组件化、服务化设计理念，分为三层：

- **控制层**：DataWorks离线加工的核心，工作流调度引擎承接着整个DataWorks的调度，包括：工作流的转实例、工作流调度，AlisaDriver主要协调、控制所有任务的执行。
- **服务层**：为应用层或外部其他应用提供服务。
- **应用层**：基于底层服务直接和用户进行交互，为用户提供可视化的操作的界面。

安全架构

DataWorks的安全架构，是由平台自身的安全实现层、平台内置的安全服务层、租户可选的安全产品层构成：

- **平台自身的安全实现层**：保障平台在代码实现和部署配置时产品自身的安全性。
- **平台内置的安全服务层**：为租户和其用户提供平台基础性的安全服务能力，如：租户资源隔离、身份认证、权限鉴别和日志合规审计等。

- **租户可选的安全产品层**：为租户和其用户提供可选的、已集成的安全产品或工具，帮助租户根据其自行定义的安全策略对其拥有的系统、数据进行安全防护和运维管理。

多租户模型

DataWorks拥有自己的多租户权限模型：

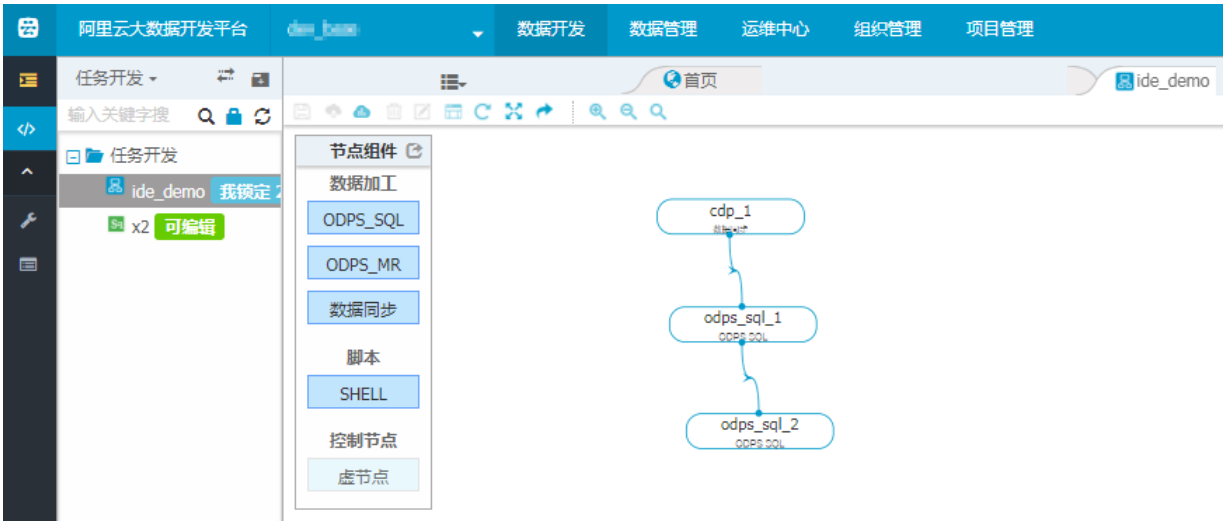
- 弹性的存储和计算资源：租户可按需申请资源配额，独立管理自己的资源。
- 租户独立管理自有的数据、权限、用户、角色，彼此隔离，以确保数据安全。

7.4 功能描述

7.4.1 数据开发IDE

DataWorks的数据开发IDE模块，提供一站式的集成开发环境，可满足大数据环境下的快速数仓建模、数据查询、ETL开发、算法开发等需求，并提供多人在线协同开发、文件版本控制等功能。

图 7-3: 数据开发



功能特性：

- 提供可视化工作流程设计器功能，类似Kettle的工具，支持用户对流程进行设计并编辑，对流程中的每一个任务节点进行相应的开发工作。
- 提供本地数据上传功能，支持本地文本数据快速上云。
- 提供海量异构数据源的数据快速集成能力。

> **说明**：
>
> 目前数据同步任务支持的**数据源类型**包括：

- 取数据支持的数据源：mysql、oracle、sqlserver、postgresql、rds、drds、maxcompute、ftp、oss、hdfs、dm、sysbase。
- 写数据支持的数据源：mysql、oracle、sqlserver、postgresql、rds、drds、maxcompute、ads、ocs、oss、hdfs、dm、sysbase。

- 提供Web IDE编程和调试环境，支持多种程序类型： SQL、MR、SHELL（有限支持）、数据同步等。
- 跨项目发布：快速将任务及代码部署到其他项目的调度系统。
- 协同开发：代码版本管理，多人协同模式下的代码锁管理和冲突检测机制。
- 提供MaxCompute（原ODPS）表搜索、资源搜索引用、自定义函数搜索引用、数据查询功能，您可轻松索引数据。

7.4.2 数据管理

数据管理为用户提供租户范围内数据表搜索、数据表详情查看、数据表权限管理、收藏数据表等功能。详细操作请参见：《*DataWorks*用户指南中的数据管理》。

7.4.3 调度系统

离线调度系统为用户提供百万量级任务的离线调度服务，并提供可视化运维界面、在线日志查询、监控告警等功能。详细操作请参见：《*DataWorks*用户指南》。

功能特性：

- 调度系统可支撑的job数量达到百万级。
- 执行框架采用分布式架构，并发作业数可线性扩展。
- 支持多时间粒度的调度周期：分钟、小时、日、周、月、年。
- 支持节点空跑、暂停、一次性运行等特殊状态控制。
- 可视化展示调度任务DAG图，极大地方便用户对线上任务进行运维管理。
- 支持实时任务运行状态监控告警功能，短信、邮件的告警方式。
- 支持单任务重跑、多任务重跑、结束进程、置成功、暂停等线上运维操作功能。
- 支持补数据（串行执行多周期实例）。
- 提供全局的任务统计信息汇总界面，任务统计内容包括：总调度任务数、出错调度任务数、运行调度任务数、计算资源消耗Top10调度任务、计算时间消耗Top10调度任务、任务类型分布等信息。

7.4.4 数据集成

提供多种异构数据源的快速集成服务，为跨平台的异构数据提供快速数据整合的能力。

功能特性：

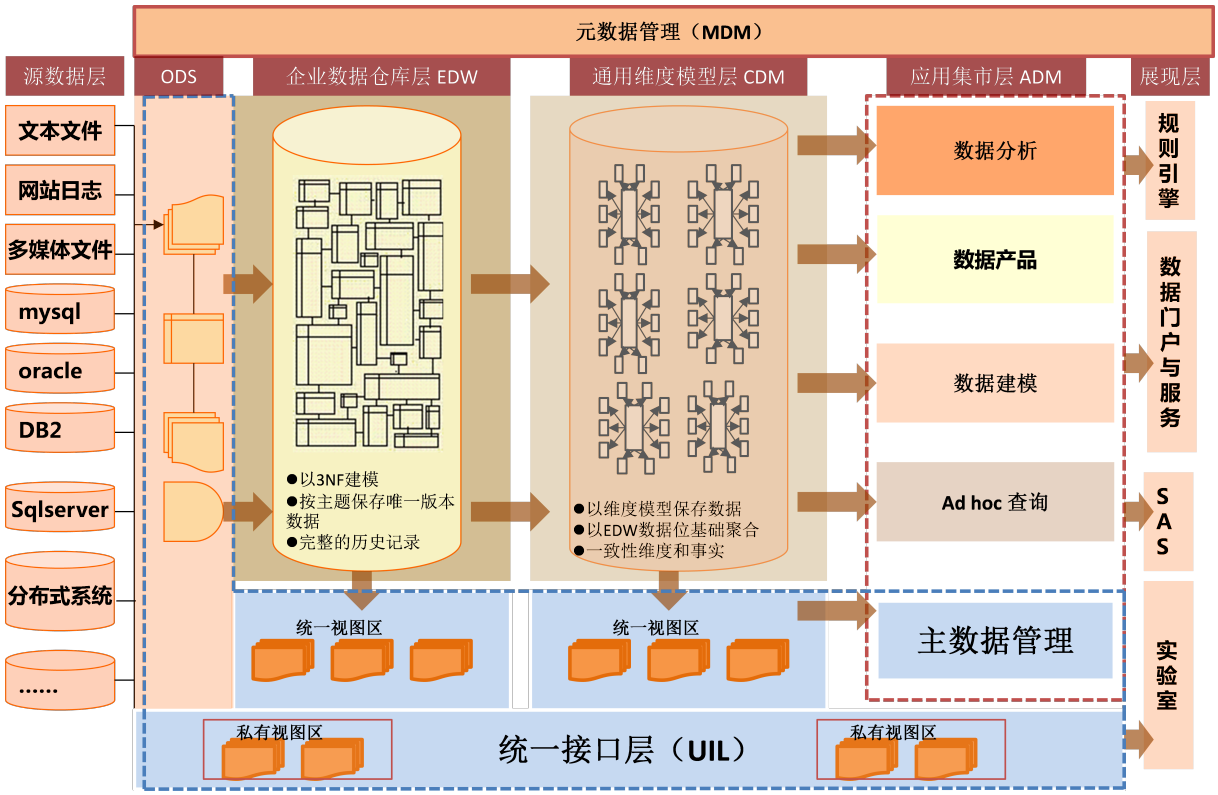
- 支持以多种数据通道（并且持续增长中）
  - 取数据支持的数据源：mysql、oracle、sqlserver、postgresql、rds、drds、MaxCompute、ftp、oss、hdfs、dm、sysbase。
  - 写数据支持的数据源：mysql、oracle、sqlserver、postgresql、rds、drds、MaxCompute、ads、ocs、oss、hdfs、dm、sysbase。
- 可靠的数据质量
  - 完美支持各种数据类型的转换，保证不失真。
  - 能精确识别脏数据，进行过滤、采集、展示，为用户提供可靠的脏数据处理，让用户准确把控数据质量。
  - 提供作业全链路的流量、数据量、脏数据探测和运行时汇报。
- 强劲的传输速度
  - 极致优化的单通道插件性能，单进程一定能够打满单机网卡（200MB/s）。
  - 全新的分布式模型，吞吐量无限水平扩展，我们能够提供GB级、乃至于TB级数据流量。
- 友好的控制体验
  - 精确且强大的流控保证，支持通道、记录流、字节流三种流控模式。
  - 完备且健全的容错处理，能够做到线程级别、进程级别、作业级别多层次局部/全局的重试。
- 清晰的内核设计
  - 专家级的框架设计经验，执行引擎更加强大，内核可以仅仅修改配置即可完成升级。
  - 更加清晰易用的插件接口，让插件开发人员专注于业务开发，而不再关注框架细节。

7.5 应用场景

7.5.1 大型数据仓库搭建

大型企业可在私有云环境下使用DataWorks来构建超大型的数据仓库。

图 7-4: 构建数据仓库



DataWorks为这类客户提供卓越的海量数据集成能力：

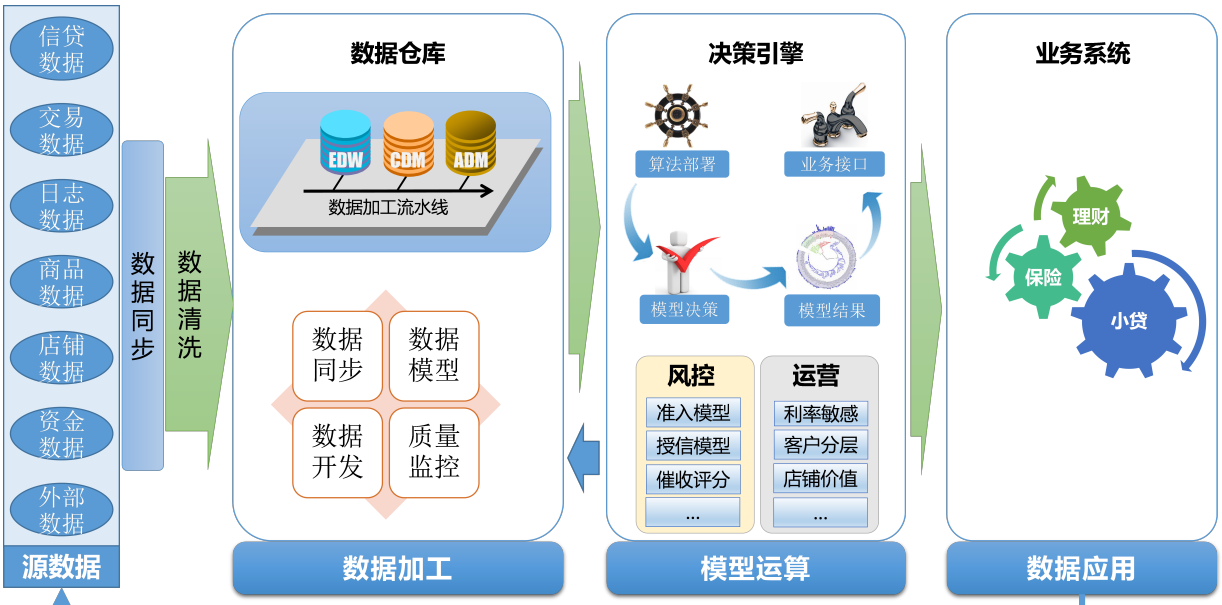
- 海量存储：可支持PB、EB级别的数据仓库，存储规模可线性扩展。
- 数据集成：支持多种异构数据源的数据同步和整合，消除数据孤岛。
- 数据开发：基于MaxCompute（原ODPS）的大数据开发，支持SQL、MR等编程框架，以及贴近业务场景的白屏化工作流设计器。
- 数据管理：基于统一的元数据服务来提供数据资源管理视图，以及数据权限审批流程。
- 离线调度：可以提供多时间维度的周期性调度能力，支持每天百万级的调度并发，并对任务调度实时监控，对错误及时告警。

7.5.2 数据化运营

- 创新业务：通过数据挖掘建模和实时决策系统，将大数据加工结果直接应用于业务系统。
- 中小企业：基于DataWorks平台快速使用和分析数据，助力企业的经营决策。

以下展示阿里小贷的数据业务运营模式：

图 7-5: 数据化运营

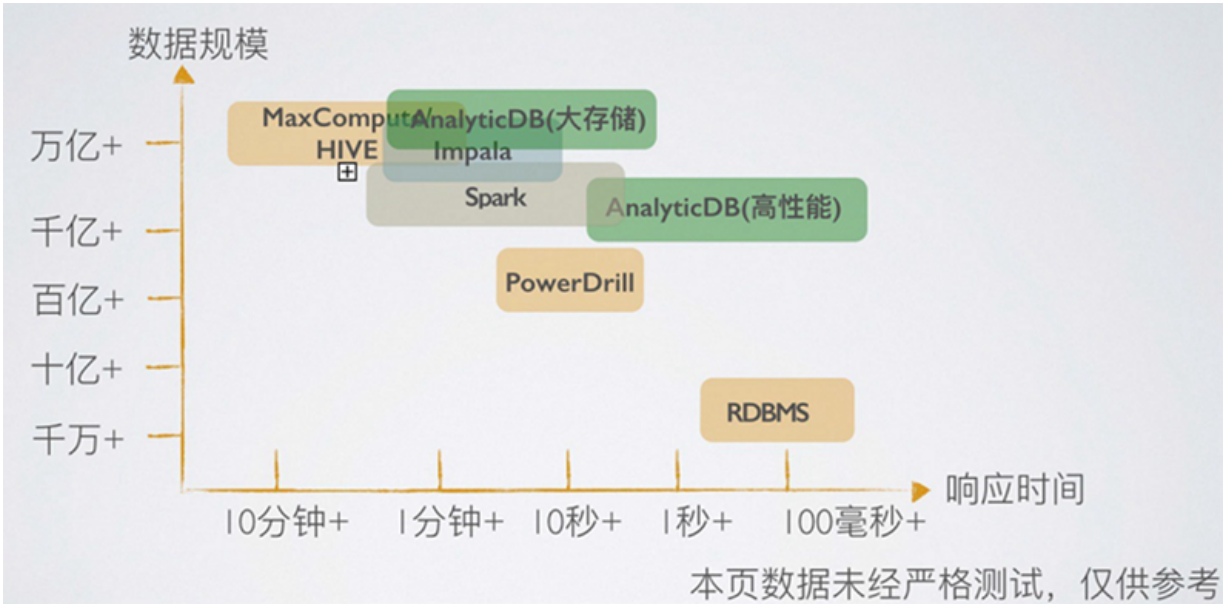

信贷数据
交易数据
日志数据
商品数据
店铺数据
资金数据
外部数据
源数据
数据同步
数据清洗
数据仓库
EDW
CDM
ADM
数据加工流水线
数据同步
数据模型
数据开发
质量监控
数据加工
决策引擎
算法部署
业务接口
模型决策
模型结果
风控
准入模型
授信模型
催收评分
...
运营
利率敏感
客户分层
店铺价值
...
模型运算
业务系统
理财
保险
小贷
数据应用
数据回流，形成闭环

8 分析型数据库

8.1 什么是分析型数据库

根据IDC 2013年发布的数字宇宙研究报告（Digital Universe）显示，在接下来的8年中，我们所产生的数据量将超过40ZB（泽字节）。作为大数据特征中最重要的Volume（容量）、Velocity（数据生产速度）的两个原始特征都在发生急剧变化，使得数据处理从业务系统的一部分演变得愈发独立，企业需要加速数据分析和挖掘过程，并由报表展现为主到强调数据洞察转型，让数据直接快速产生价值（Value）。在业务系统中，我们通常使用的是OLTP（OnLine Transaction Processing，联机事务处理）系统，如MySQL，Microsoft SQL Server等关系数据库系统。这些关系数据库系统擅长事务处理，在数据操作中保持着严格的一致性和原子性，能够很好支持频繁的数据插入和修改，但是，一旦需要进行查询或计算的数据量过大，达到数千万甚至数十亿条，或需要进行的计算非常复杂的情况下，OLTP类数据库系统便力不从心了。

图 8-1: 产品对比图



分析型数据库（Analytic DB，原ADS）和主流数据系统进行对比（数据未经严格测试，仅供参考）时，便需要OLAP（On-Line Analytical Processing，联机分析处理）系统进行处理。从广义上，OLAP系统是针对OLTP系统而言，并不特别关心对数据进行输入、修改等事务性处理，而是关心对已有大量数据进行多维度的、复杂的分析的一类数据处理系统。在具体的产品中，通常将OLAP系统分为MOLAP、ROLAP和HOLAP三类。

多维OLAP（Multi-Dimensional OLAP，简称 MOLAP），是预先根据数据待分析的维度进行建模，在数据 的物理存储层面以“立方体”（Cube）的结构进行存储，具有查询速度快等优点，但是数据必须预先建模，无法依据使用者的意愿进行即时灵活的修改。而关系型OLAP（Relational OLAP，简称ROLAP），则使用类似关系数据库的模型进行数据存储，通过类似SQL等语言进行查询和计算，优点是数据查询计算自由，可以灵活的根据使用者的要求进行分析，但是缺点是在海量数据的情况下分析计算缓慢。至于HOLAP，则是MOLAP和ROLAP的混合模式。

而分析型数据库，则是一套RT-OLAP（Realtime OLAP，实时OLAP）系统。在数据存储模型上，采用自由灵活的关系模型存储，可以使用SQL进行自由灵活的计算分析，无需预先建模，利用分布式计算技术，分析型数据库可以在处理百亿条甚至更多量级的数据上达到甚至超越MOLAP类系统的处理性能，真正实现百亿数据毫秒级计算。

分析型数据库让海量数据和实时与自由的计算可以兼得，实现了速度驱动的大数据商业变革。一方面，分析性数据库拥有快速处理迁移级别海量数据的能力，使得数据分析中使用的数据可以不再是抽样的，而是业务系统中产生的全量数据，使得数据分析的结果具有最大的代表性。更重要的是，Analytic DB采用分布式计算技术，拥有强大的实时计算能力，通常可以在数百毫秒内完成百亿级的数据计算，使得使用者可以根据自己的想法在海量数据中自由的进行探索，而不是根据预先设定好的逻辑查看已有的数据报表。

更加重要的是，由于分析型数据库能够支撑较高并发查询量，并且通过动态的多副本数据存储计算技术来保证较高的系统可用性，所以能够直接作为面向最终用户（End User）的产品（包括互联网产品和企业内部的分析产品）的后端系统。如淘宝数据魔方、淘宝指数、快的打车、阿里妈妈达摩盘（DMP）、淘宝美食频道等拥有 数十万至上千万最终用户的互联网业务系统中，都使用了分析型数据库。

分析型数据库作为海量数据下的实时计算系统，给使用者带来极速自由的大数据在线分析计算体验，最终期望为大数据行业带来巨大的变革。

8.1.1 存储模式

在0.9版本中，分析型数据库拥有两种存储模式，对应不同的成本和业务模型：

高性能存储模式实例：使用全SSD（或Flash卡）进行计算用数据存储，使用内存作为数据和计算的动态缓存的实例。可在双千兆或双万兆网络服务器上良好运行，具有计算性能好、查询并发能力强的优点，缺点是存储成本较高。

大存储模式实例：采用SATA磁盘进行分布式存储，作为计算用数据存储，使用SSD和内存两级作为数据和计算的动态缓存的实例。必须在双万兆网络服务器上才能良好运行，具有存储成本低的优点，但查询并发能力相对较弱，一次性计算较多行列时性能较差。

您创建数据库实例时，可以选择使用的存储模式，一经选定无法中途更改。

8.1.2 系统资源管理

分析型数据库通过ECU（弹性计算单元）进行资源管理。通过操作系统底层技术和飞天提供的分布式资源调度能力，分析型数据库为每个数据库实例创建完全独立的FrontNode、ComputeNode、BufferNode进程。每个数据库至少拥有FrontNode、ComputeNode、BufferNode进程各两个（双副本双活）。

您可以通过控制ECU型号，来决定FrontNode、ComputeNode、BufferNode进程的配置，通过ECU型号可以区分的资源包括CPU 核数（支持独占和共享）、内存大小（独占）、SSD大小（独占）、网络带宽（独占）、SATA数据逻辑大小（仅大存储实例的Compute Node可选）。

您可以通过控制ECU的数量，来决定一个数据库实例所启用的ComputeNode数量，从而通过ECU类型上所配置的比例来按比例启动若干个FrontNode和BufferNode，从而达到容量的水平伸缩的目的。

FrontNode、ComputeNode、BufferNode进程可以混部在同一批物理机（默认），也可以配置参数强制不同的角色在不同的物理机上运行。

除此之外，分析型数据库的后台任务、数据库AM等也会占用一定量的系统资源。

8.1.3 计算引擎

在最新的分析型数据库版本中，拥有两套计算引擎。

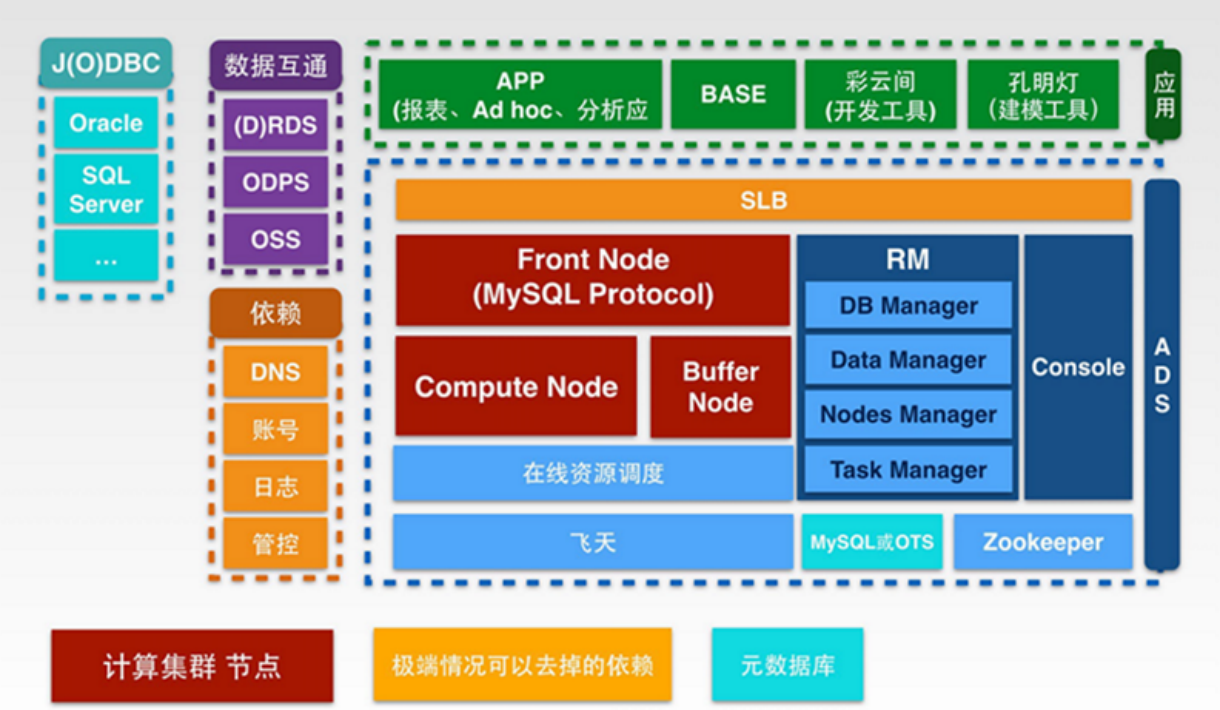
Compute NodeLocal/Merge（简称LM）：先前版本的旧计算引擎，优点是计算性能很好，并发能力强，缺点是对部分跨一级分区列的计算支持较差。

FullMPP Mode（简称MPP）：新增的计算引擎，优点是计算功能全面，对跨一级分区列的计算有良好的支持，可以通过全部TPC-H查询测试用例（22个），和60%以上的TPC-DS查询测试用例。缺点是计算性能相对LM引擎较差，并且计算并发能力相对很差。

在开启Full MPP Mode引擎功能的数据库中，分析型数据库会自动对查询Query进行路由，将LM引擎不支持的查询路由给MPP引擎，尽可能兼顾性能和通用性，用户也可以通过Hint自行决定某个Query 使用什么样的计算引擎。

8.2 产品架构

图 8-2: 产品架构图

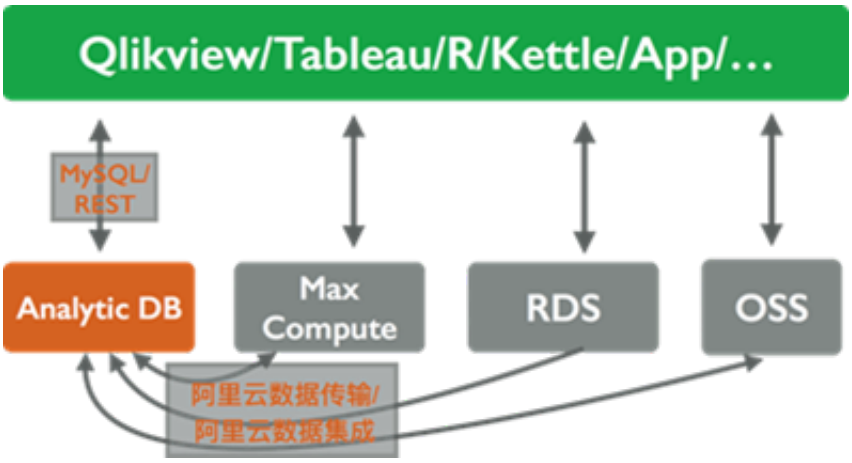


Analytic DB（原 ADS）是构建在阿里云分布式操作系统飞天之上的基于MPP架构并融合了分布式检索技术的分布式实时计算系统。如图所示，Analytic DB的主体部分主要由四个部分组成。

- **底层依赖**：包括用于进行资源虚拟化隔离、数据持久化存储、构建数据结构和索引而使用的阿里云飞 天分布式操作系统套件，用于存储分析型数据库的各类元数据（注意并不是实际参与计算用的数据）的阿里云RDS关系数据库或阿里云表格存储，以及用于各个组件间进行分布式协调的开源Apache ZooKeeper模块；
- **计算集群**：是计算资源实际包括的内容，均可进行横向扩展。包括用于处理用户连接接入认证、鉴权、查询路由与分发以及提供元数据查询管理服务的Front Node、用于进行实际的数据存储与计算的Compute Node、用于处理数据实时更新数据缓冲和实时数据写入版本控制的Buffer Node。计算集群运行在阿里云超大规模分布式操作系统飞天上，并通过在线资源调度模块来使用飞天调度计算资源；
- **控制集群**：暨资源管理器RM，用于控制计算集群中数据库资源分配、数据库内数据和计算资源的分布、管理飞天集群上的计算节点、管理数据库后台运行的任务等。控制集群实际上由多个模块组成，一个控制集群可以同时管理位于不同机房部署的多套计算集群；

- **外围模块**：如用于管理Front Node的分组和负载均衡的阿里云负载均衡、用于发布数据库域名的阿里云DNS系统、阿里云账号系统、Analytic DB的控制台（Admin Console）和用户控制台（DMS for Analytic DB）等。

图 8-3: 系统交互



在对外部系统的交互上，分析型数据库能够从MaxCompute批量导入数据，并且可以快速批量导出海量数据到MaxCompute；可以实时的将RDS的数据同步到分析型数据库中（需借助外部同步工具）；计划在0.10可以通过CDP从OSS拉取数据并快速批量导出海量数据到OSS。

对于前端业务，分析型数据库允许任何遵循MySQL 5.1/5.5/5.6系列协议的客户端和驱动进行连接。例如：MySQL 5.1.x jdbc driver、MySQL 5.3.x odbc connector(driver)、MySQL 5.1.x/5.5.x/5.6.x客户端、 java、python、C/C++、Node.js、PHP、R（RMySQL）、Websphere Application Server 8.5、Apache Tomcat、JBoss等。

8.3 功能特性

8.3.1 实体

8.3.1.1 用户

- 通过云账号进行登录。
- 不同用户可以被授予不同的权限。
- 用户的操作均可被细粒度审计。

8.3.1.2 数据库

- 租户隔离的基本单位。
- 可基于数据库对计算资源进行物理或虚拟化隔离。

- 可基于数据库进行大量的系统配置。
- 数据库的创建用户为数据库的owner或数据库管理员。

8.3.1.3 表组

- 资源分配的最小单元。
- 可进行副本数、分片数、超时时间等多种配置。
- 同表组内的表可进行Join加速（维度表可和任何表组的表进行Join加速）。

8.3.1.4 事实表

- 支持标准的关系表模型。
- 表支持多级分区以及多种分区类型。
- 支持根据若干列进行数据聚集，以实现高性能查询优化。
- 单表支持最大1024列。
- 单表可支持数千亿行甚至更多的数据。

8.3.1.5 维度表

- 支持和任意表组的任意表以任意列进行Join加速。
- 最大可支持千万级的数据条数。
- 无需指定分区方式。

8.3.1.6 列

- 支持boolean、tinyint、smallint、int、bigint、float、double、varchar、date、timestamp等多种MySQL标准数据类型；
- 分析型数据库特有类型多值列multivalue，高性能存储和查询一个列中的多种属性值信息；
- 可针对列配置不建立自动化索引，0.8版本前支持追加建立HashMap索引，0.9版本无需手动追加建立HashMap索引。

8.3.1.7 ECU

ECU是弹性计算单元，是Analytic DB的资源调度和计量的基本单位。

ECU可配置不同的型号，每种型号的ECU可配置CPU核数（最大、最小）、内存空间、磁盘空间、网络带宽等多种资源隔离指标，Analytic DB出厂时已根据机型配置预设了多种最佳的ECU型号Front Node、Compute Node、Buffer Node均由ECU进行资源隔离，Compute Node的ECU数

量和型号需由用户配置（弹性扩容/缩容），Front Node、Buffer Node的数量和型号系统自动根据ComputeNode的情况换算和配置。

8.3.2 DDL

数据库管理：

- 通过DDL创建数据库；
- 通过DDL删除数据库；
- 看全部有权限的数据库列表（show databases）；
- 查看和管理每个数据库的访问信息（域名、端口等信息）；
- 通过DDL对数据库使用的ECU资源进行扩容、缩容操作3.2.2表组和表管理；
- 通过DDL创建表组和修改表组属性；
- 通过DDL创建表；
- 通过DDL在已创建的表中增加列；
- 通过DDL修改表属性；
- 通过DDL修改索引；
- 支持Create-table-as-Select创建临时表。

8.3.3 DML

8.3.3.1 SELECT

- 和标准MySQL的Query兼容达90%；
- 支持表达式、函数、别名、列名、case when等列投射形式；
- 支持From表名as别名，Join表名as别名；
- 支持事实表之间的Join（若需加速Join则有限定条件）和事实表与维度表的Join（几乎无限制）；
- 支持多个on条件的Join（若需加速Join则其中必须包含一个一级分区列）；
- 过滤条件（where）中，支持and和or表达式组合、支持函数表达式、支持between、is等多种逻辑判断和条件组合；
- 支持多列group by，并且支持case when等列投射表达式产生的别名进行group by，支持常见的聚合函数；
- 支持order by表达式、列，并支持正序和倒序；
- 支持having；

- 支持子查询（建议不超过3层），支持在特定条件下的两个子查询的Join，支持过滤条件中的in中使用数据全部源自维度表的子查询，通过Full MPP Mode支持任意的in子查询；
- 支持带有一级分区列的多列的[count]distinct，在Full MPP Mode下支持任意列的[count]distinct；
- 支持常数列；
- 支持union/union all，有限定条件的支持minus/intersect。

8.3.3.2 INSERT/DELETE

- 支持对已定义主键的实时写入表进行INSERT、DELETE操作。
- 在资源足够的情况下，单表可支撑5万次每秒以上的INSERT操作，数据插入后数分钟内生效。
- 多种机制保障写入成功的数据不会丢失insert支持overwrite、ignore两种模式。
- 支持insert into...select from。

8.3.4 权限与授权

授权模型如下所示。

- 支持标准MySQL模式的权限模型。
- 可以对数据库、表组、表、列四个级别进行ACL授权。
- 支持数据库owner授权给任意合法账号。
- 支持可创建数据库权限单独控制或外挂（目前只支持UMM）控制。
- 支持超级管理员、系统管理员等角色。
- 支持每个级别授予不同的权限。
- 支持add user/remove user。
- 支持grant语句进行授权。
- 支持revoke语句进行权限回收。
- 支持show grants on语句查看各级对象上的用户权限。
- 支持list users查看全部有权限的用户。
- 超级管理员：集群初始建立时指定的账号，具有任命系统管理员和数据库管理员的权限，无其他权限。
- 系统管理员：由超级管理员任命，具有查看和操作SYSDB的权限。
- 数据库管理员：由超级管理员任命，具有为其他用户创建数据库和删除其他用户的数据库的权限。

- 数据库Owner：数据库的所有者，具有一个数据库的全部权限，并可以授权一般用户访问自己的数据库。

8.3.5 Data Pipeline

海量数据快速导出。

- 支持任何SELECT语句的查询输出。
- DUMP DATA可以将大量数据快速导出到TFS（对内）、OSS（对外，暂未上线）等DFS中以及MaxCompute。
- DUMP DATA性能可达到1000万条数据10s内导出完毕3.5.2数据的导入。
- 支持类BULKLOAD模式导入MaxCompute /OSS/RDS中用户存放的数据。
- 内置支持使用LOAD DATA命令进行导入。
- 内置支持导入数据owner校验，保证导入安全。

8.3.6 特色功能

8.3.6.1 特色函数

- 支持高性能的根据地理坐标范围筛选数据（方形和圆形圈选、点距离计算等）。
- 支持智能分段统计函数（指标的自动分段）。
- 支持快速多列聚合函数。
- 对全部列根据列的数据分布智能建立索引，无需您进行任何操作。
- 对完全不需进行检索的列，您可手动关闭智能索引。

8.3.6.2 智能缓存和CBO优化

- 拥有多层智能缓存，最大限度的利用内存加速计算，但是可计算的数据量大小不受内存大小限制。
- 拥有智能的CBO优化器，可以根据数据分布情况和您的SQL执行情况动态优化计算执行计划，让您从SQL写法优化中解脱。
- 拥有智能的分布式长尾处理技术，大幅度降低分布式系统中单节点繁忙以及网络等不确定性因素对响应时间的影响。

8.3.6.3 Quota控制

- 支持每次导入数据量、单表导入次数、并发任务数、DB导入次数、DB导入总数据量等控制。
- 支持ECU总数据量控制、ECU库存控制、单次扩容ECU数量控制、每天扩容次数控制。

- 支持DB的总表数、总实时表数、总表组数、单表最大列数、单表分区数（一级、二级）控制。
- 支持系统元数据库（SYSDB、ADMIN DB）流控和风控。
- 支持大存储模式实例，使用SATA进行计算数据存储，使用SSD和内存作为缓存加速热点数据查询。

8.3.6.4 Hint和小表广播

- 支持通过Hint干预执行计划，如计算引擎的选定和索引使用的控制。
- 支持小表广播模式Join，在小表（物理表或虚表）Join大表时通过 Hint指定小表广播，在不符合加速Join条件时亦可获得比较好的Join性能。

8.3.7 元数据

8.3.7.1 information_schema

- 最大限度兼容MySQL标准的db、表、列等信息，并且元数据完全可被您使用并可进行交互。
- 数据导入的记录和进度均可在元数据库进行查询。
- 提供ECU运行状态，以及ECU扩容缩容记录表。
- 元数据按照DB进行隔离，您无法访问无权使用的元数据。

8.3.7.2 performance_schema

- 提供实时元仓，可进行SQL粒度的查询审计以及分钟粒度的插入性能统计。
- 提供分钟级别更新的QPS、RT、请求数、数据量大小等实时性能监测。
- 元数据按照DB进行隔离。

8.3.7.3 sysdb

- 面向系统管理员和运维人员的元数据库。
- 可以观察Analytic DB全部模块的运行状态、运行历史记录等，拥有数十张各个主题的系统元数据表。
- 可以查看系统运行的运行时状态，并在有需要时可以进行修改。
- 可以查看或修改系统各组件的参数，运行计算集群升级、降级、扩容、缩容、挂起命令。

8.3.8 管理控制台

8.3.8.1 用户控制台（DMS for Analytic DB）

- 支持阿里云账号登录与鉴权。

- 提供图形化的数据库创建与管理、表组/表的创建与管理功能。
- 提供友好的SQL查询调试功能，并提供图形化的执行计划展示。
- 提供友好的用户与权限查看、管理功能。
- 提供友好的数据导入导出运行界面以及查看状态与进度的界面。
- 提供两分钟内实时系统性能报告的展示以及最近七天内小时粒度详细的离线性能报告展示。
- 提供DB资源可视化。
- 提供扩容、缩容、查看扩容/缩容状态和历史的功能。

8.3.8.2 运维管理控制台（Admin Console、Tesla）

- 用于系统后台运维人员管理系统资源、监控系统运行状态和修改系统参数。
- 提供图形化的界面展示各个DB的存储计算资源，以及在物理节点上的占用、分布。
- 提供图形化的查看和管理系统参数功能。
- 提供图形化的查看和管理数据导入全链路状态功能。
- 提供白屏化的分布式日志提取和查看工具。

8.4 系统架构简介

8.4.1 系统架构概况

Analytic DB（原ADS）是基于飞天的分布式数据库，可针对海量数据进行实时计算。

它的主要组成包括飞天，FuxiService和Analytic DB，其中飞天负责管理整个集群的物理资源，并为Analytic DB提供数据的持久存储功能，FuxiService负责Analytic DB所申请的资源。

8.4.2 飞天

8.4.2.1 飞天模块简介

飞天系统是阿里云自主研发的分布式计算平台，可为用户提供分布式文件存储和分布式调度功能。飞天主要包括以下几个模块：

盘古

盘古是基于AliOS的可扩展的分布式文件系统，是产生和处理数据的存储平台，可运行在成本相对较低的硬件上同时具有容错能力并且可保证大量客户端的高性能访问。

盘古集群包括三台master和多台chunkserver，其中master是热备，同一时间只有一台在工作。盘古上的文件会被切成多个固定大小的存储单元，每个存储单元都叫一个chunk。存储数据块chunk的

机器称为chunkserver，对每个数据块，创建之初，pangumaster都会分配一个128位的ID，盘古客户端根据ID来读取存储在磁盘上的数据块。

盘古master维护了整个文件系统的元数据，其中包括命名空间，文件到数据块的映射及数据块的存储地址等。

作为整个分布式文件存储系统的大脑，盘古master控制着系统层面的活动如孤立数据块的垃圾回收，chunckserver间的数据合并，判断chunkserver是否健康，恢复由于chunkserver宕机所造成的数据块丢失等。盘古master同时还负责调节同一时间片中多台客户端对数据的访问来维护同一集群中数据的完整性。盘古master只对客户端提供元数据相关的操作，而数据传输相关的通讯都直接在chunkservers间进行。

为了保证可靠性，同一个数据块会被存储在多个chunkserver上，保存的份数可以通过参数来设定，目前ADB在盘古上存储数据的副本数是3份。

伏羲

伏羲是飞天的资源调度模块，其主要功能是充分利用整个集群的硬件资源来服务用户和系统的计算需求。伏羲将计算分为两类：服务（Service）和 临时任务（job）。Service是伏羲上的常驻进程，由用户申请创建及销毁，伏羲不会主动销毁Service进程。job是伏羲上的临时任务，一旦任务结束，资源就会被释放，由伏羲回收。ADS的进程在飞天上作为伏羲Serive启动。

伏羲包括两台master和多台tubo，其中master是冷备，同时只有一台在工作，而每台计算节点上都启动了一个tubo进程，用来管理单机资源，如汇总整机的可用的CPU，内存，硬盘，网络，同时记录每台机器已被占用的资源，每个机器上的tubo进程会将这些资源汇报给伏羲master，由伏羲master统一管理及调度。

女娲

女娲是阿里云飞天系统的一个基础模块，主要提供飞天系统中分布式一致性相关的服务，具有性能高和水平扩展性强的特点，女娲为飞天提供：

- 分布式锁服务（Locking Service）：分布式锁是控制分布式系统之间同步访问共享资源的一种方式。在分布式系统中，常常需要协调它们的动作。如果不同的系统或是同一个系统的不同主机之间共享了一个或一组资源，那么访问这些资源的时候，往往需要互斥来防止彼此干扰来保证一致性，在这种情况下，便可以使用Nvwa提供的Locking Service。
- 订阅通知服务（Notification Service）：每个应用可以通过向Nvwa订阅相关文件，一旦该文件发生改变，所有订阅了该文件的应用都将收到此文件被变更的通知；

- 轻量存储服务（Meta Storing）：由于Nvwa的高可靠性，一些应用将它们的key metadata存储在Nvwa上，例如，伏羲在女娲上存储了checkpoints信息，沉香在女娲上存储了整个飞天集群的配置信息等。

大禹

大禹，负责飞天基础模块的部署和配置管理，部署的含义是各个机器按照用户指定的角色信息，运行起来相应的程序。一个飞天集群中，同一台机器上可以部署多个飞天角色，例如，盘古chunkserver和fuxi的tubo是运行在同一台机器上的，chunkserver负责管理SATA盘上的数据，而tubo负责管理机器上的CPU/MEM等资源信息。在部署时，大禹就会根据用户定义的角色信息，在一台机器上同时启动chunkserver和tubo的进程。

沉香

沉香是飞天的配置管理服务，可以管理集群飞天环境的各项配置，包括集群飞天版本和各个模块的flag、conf。通过沉香可以使配置项的修改立即更新到对应角色的机器上，只是飞天进程目前不支持动态加载配置，所以需要重启飞天进程使新配置生效。所有的配置都可以在web portal上浏览修改，并且沉香保存了修改的历史记录，对整个集群的集群进行统一高效的管理。

沉香支持配置表（TABLE）和配置文件（FILE）两种配置类型，可以通过沉香的web portal上修改，目前大多数功能都可以在web上操作。沉香配置目前都存储在nvwa里，每个计算节点上chenxiang-agent会订阅Nvwa消息，一旦修改会立刻同步到相应机器上。

包管理系统

包管理系统在飞天内称为Package Manager，负责管理飞天上运行的任务所依赖的包。

中控机

中控机（Admin Gateway）简称AG，是整个飞天集群的运维管控节点，通过AG可以对整个飞天集群进行所有运维操作如进程启停，飞天任务查看，资源查看等等。大禹和沉香的管控进程都启动在AG机器上，对整个集群进行监控及管理。

8.4.2.2 飞天常用运维工具

Admin Gateway

Admin Gateway是飞天上单独的角色，其上部署了对飞天平台进行运维的所有工具，常用工具如下所示。

- pu -- pangu 文件操作：包括创建删除目录，上传下载文件等，全路径为：*/apsara/deploy/pu*，常用命令如下。

表 8-1: 文件操作常见命令

Command	Description
ls	列出指定目录下的文件及目录，必要参数PanguDir和PanguFile，例如：pu lspangu://localcluster/dir1，PanguDir默认值为pangu://localcluster
rm	pu rmPanguFile，删除盘古文件，删除的文件会放到盘古回收站中（ pangu://localcluster/deleted目录）
mv	pu mv PanguFile1Pangufile2/PanguDir，移动文件
mvdir cp	pu mvdir panguDir1panguDir2，移动目录 pu cp File1File2，文件拷贝
mkdir	pu mkdirPanguDir，在盘古上创建目录
meta put	pu metapanguFile，查看盘古文件的meta信息 pu put localfilepangufile，从本地copy文件到盘古
get	pu get pangufilelocalfile，从copy盘古文件到本地

8.4.2.3 目录结构

飞天主要角色的目录结构如下所示。

路径	角色	作用
/apsara/builds	AG	飞天安装包保存路径，飞天在安装时，会先把安装包下载到 AG的此目录下，下载完成之后，再推送到其他机器上
/apsara/chenxiang /apsara/conf	AG 所有角色	沉香agent目录，其中有沉香agent的执行脚本和日志文件 本机保存的该集群配置文件，在沉香上修改之后的配置文件 都会被沉香同步到此目录
/apsara/dayustore	所有角色	本机保存的该集群安装文件，安装或升级飞天时，会将AG上的安装包，同步到集群中每台机器上
/apsara/deploy /apsara/lib64	AG 所有角色	主要包含pu和puadmin两个命令以及两个命令的运行日志 飞天sdk目录
/apsara/nvwa	nvwa	nvwa进程根目录
/apsara/fuxi_master	fuximaster	fuximaster进程根目录

/apsara/pangu_master	panguma ster	pangumaster进程根目录
/apsara/security	所有角色	访问此飞天集群所待的秘钥文件，对于client，如果没有秘 钥，是无法访问该飞天集群的
/apsara/watch_dog	所有角色	飞天watch_dog（守护进程）进程的根目录
/apsara/tubo /apsara/pangu_chunkserver	tubo pangu chunkserver	tubo进程根目录 panguchunkserver根目录
/apsara/TempRoot	tubo	飞天job执行日志目录

8.4.2.4 各进程作用

整个飞天集群由多个角色组成，对应每个角色，都会启动一个单独进程。

- fuximaster

  /apsara/fuxi_master/fuxi_master fuximaster 机器上，fuximaster进程。

- pangumaster

  /apsara/pangu_master/pangu_master，pangumaster机器上的pangumaster进程。

- nvwa

  /apsara/nuwa/java4zk，master上nvwa进程。

- package manager进程

  /apsara/package_manager/package_manager，packagemanager进程。

- tubo

  /apsara/tubo/tubo，tubo进程。

- panguchuckserver

  /apsara/pangu_chunkserver/pangu_chunkserver，盘古chunkserver进程。

- watchdog

  /home/tops/bin/python/apsara/watch_dog/watchdog.py，watchdog进程，是其他飞天进程的守护进程，开机时自启动，在其他进程异常退出时负责重新拉起其他进程。

8.4.2.5 查找和阅读日志

Analytic DB 运维中，最常用的飞天日志是 job 运行过程中产生的日志，日志在 tubo 机器上的 /apsara/TempRoot 目录下，通过以下事例来演示如何查看该日志。例如，在生产环境中发现 Analytic DB 中某个实时表的 build 任务失败，此时需按照以下步骤排查问题。

1. 登录飞天 Admin Gateway（AG）。
2. r wl 找出 progress 是**Failed**的 job，如下所示。

```
"ads/job1445590217609_0_second_20151023165355339_220243": {
        "accessName": "nuwa://ATADS-DTDREAM:10240/ads/job1445590217609_0_s
econd_20151023165355339_220243/JobMaster",
        "account": 0,
        "jobMasterLaunchTime": 0,
        "lastStatusChangeTime": 0,
        "lastUpdateTime": "Fri Oct 23 19:30:45 2015\n",
        "progress": "Failed",
        "replyAddress": "",
        "resourceUsage": {}},
```

其中第一行 ads/job1445590217609_0_second_20151023165355339_220243 即为任务名称（job_name）。

3. r status job_name 得到任务的详细信息。

```
r jstatus adshz1/job1446803647732_0_second_20151106175421109_38845|grep
workerAddress
 "workerAddress": "tcp://xxx.xxx.xxx.xxx:54445",
"workerAddress": "tcp://xxx.xxx.xxx.xxx:46540",
```

从以上结果可以得到执行这个任务的 ip 列表。

4. 登录上步中得到的任何一个 ip，进入 */apsara/TempRoot/job_name*目录中查看 stderr.out 日志，或直接 `grep -r -i error ./*`，即可获取任务失败原因。

8.4.3 FuxiService

FuxiService是Fuxi提供的在线服务的调度模块，FuxiService负责在线服务的资源申请，所申请资源Worker的全生命周期管理（包括Worker的拉起，停止，重启，销毁），以及Worker的Failover处理。

8.4.3.1 FuxiService基本架构

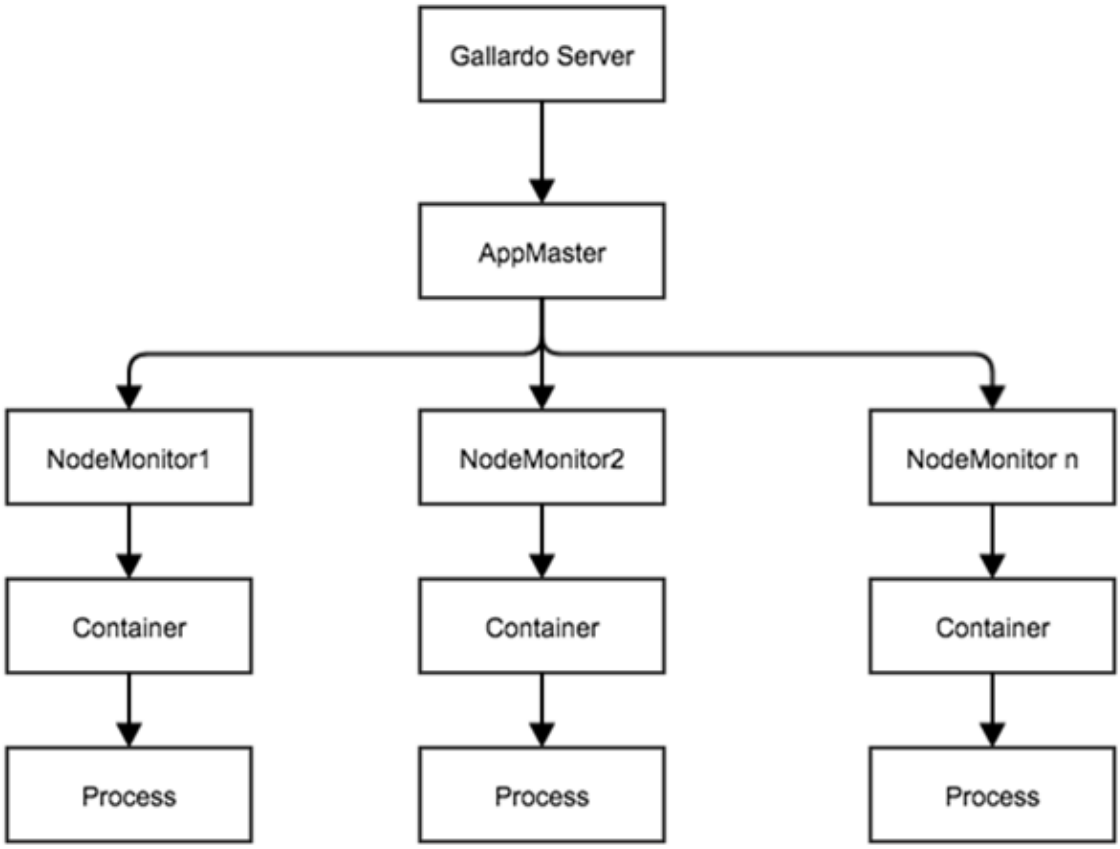
目前Analytic DB有三个模块由飞天拉起，分别是COMPUTENODE，FRONTNODE，BUFFERNODE，每个DB的每个模块在FuxiService中都是一个App，一个App下可以有多个进程。由FuxiService统一对这些App下的进程进行资源的申请，管理及释放等。

FuxiService主要有Server，UI，RMUI，Rm， Nm， Am和Container六个模块组成。

- Gallardo Server：提供http接口，接收ADS 的资源申请，管理（启动/暂停），释放请求。
- Gallardo UI：为ADS提供所创建App列表以及App下所有Worker 的详细信息（例如： Host，Pid 等）。
- RMUI：FuxiService的运维管控平台，通过多种维度展示FuxiService所管理在线服务申请的资源以及所托管的App及App下的进程。
- RM：负责执行在线服务的资源分配，以及为在线服务提供各种复杂的调度策略。
- NM：负责单机在线服务具体 Worker 的资源隔离和共享以及日志收集等功能。
- AM：负责单个 App资源的申请，释放，以及Worker的Failover 策略（例如：宕机迁移）。
- Container：负责单机在线服务具体 Worker的生命周期管理以及Failover 策略（例如：本地容灾）的执行。

基本工作流程如图 *8-4:* 工作流程图所示。

图 8-4: 工作流程图



1. Gallardo Server接收ADS 提交的资源申请请求。
2. 对每个 App，FuxiService都启动一个AppMaster进程，负责此App下的资源管理。

3. AppMaster向fuxi提交资源申请，得到可用机器后，在每个可用机器上，都启动一个对应此App的NodeMonitor 进程。NodeMonitor 在该机器上启动。

8.4.3.2 查找和阅读日志

Fuxi-Service（原Gallardo）所有日志汇总。

- Fuxi-Service Rm 相关问题：资源无法分配。日志路径定位。
  1. 在AG上执行r wl 命令，查看名称为garuda/garudaAppMaster的Job所在的机器。
  2. ssh 到上一步中找到的机器，Fuxi-Service Rm的日志为*/apsara/TempRoot/garuda/garudaAppMaster/logs/hadoop-admin-resourcemanager-*.log*。
- Fuxi-Service Nm相关问题：暂时不用关注。日志路径定位。
  1. ssh到Nm所在机器，jps查看进程名为NodeMoniTor的java进程的pid。
  2. 执行cd /proc/pid/cwd/logs/ ，日志为*hadoop-admin-resourcemanager*.log*。
- Fuxi-Service Am相关问题：创建DB失败、提交Batch 失败、Worker 拉不起来或者Worker 启停更新等操作失败。
  1. 打开当前集群的fuxi index页面，单击第一个超链接scheduler，找到对应名称 App所在的address一列的ip 地址。
  2. ssh 到上一步中找到的机器。
  3. 执行cd /home/admin/gallardo-am-logs 命令。
  4. 执行 for f in ls -rt; do echo $f; grep ${appName}命令，appName为所要查看的App名称。

     执行完成后，显示的最后一个统计结果大于 0 的日志就是目标日志。
- Fuxi-Service Container 相关问题：Worker 拉不起来或者Worker 启停更新等操作失败。
  1. 打开所需查看的App对应的Task 明细页面，例如*http://xxx/task?appname=$%7BappName%7D&group=rmscheduler&pool=$%7BappName%7D*。
  2. 查找对应Worker所在的机器Host。
  3. ssh 到Worker所在机器。
  4. 执行cd /home/admin/gallardo-container-logs命令。
  5. 执行ll -rt | grep task_name_suffix命令。

     其中，task_name_suffix为TaskName横杠分割的后4位，TaskName为在第二步页面中对应的TaskName列。

示例：TaskName为new_dmp_5-BUFFERNODE-2-841-cm1-1，则${task_name_suffix}为2-841-cm1-1。

上述命令执行的最后一个日志即为Contanier对应日志。

- Fuxi-Service Server相关问题：App/Worker创建/Worker申请失败/home/admin/install/gallardo-server/logs/gallardo-server.log。
- Fuxi-Service UI 相关问题：App/Worker相关信息查询失败/home/admin/install/gallardo-ui/logs/gallardo-ui.log。

8.4.3.3 FuxiService运维工具

RMUI 是FuxiService的运维页面，在专有云和一体机中，RMUI都部署在Analytic DB的中控机AG上，访问地址为http://agip:8315/index ，进入页面后会看到三大部分。

图 8-5: RMUI

clusterName	scheduler	resourceManager	systemLevel	amUi	amApplication
ATADS-DTDREAM	scheduler	resourceManager	systemLevel	AmApp	not set

clusterName：此飞天集群的集群名scheduler，此页面主要显示目前FuxiService已经启动或将要启动的App及App所分配的资源。

8.4.3.4 分析型数据库主体

Analytic DB 本身是个分布式数据库，可对数据进行增删改查等操作，与传统数据库相比，还可跟阿里云的 MaxCompute（原 ODPS） 进行数据互通，可以从 MaxCompute批量导入数据，分析型数据库内部模块如下。

- ResourceManager：资源控制节点，控制数据导入，上下线等过程，并为计算节点分配数据 shuffle 任务。
- Builder ：数据构建节点，管理在 pangu 或 MaxCompute上的数据构建过程，实时监控任务状态。
- FrontNode：数据查询节点，负责 SQL 解析和分发，汇总计算结果，进行权限管理等，是用户访问 Analytic DB 的入口。
- ComputeNode：计算节点，负责管理 Analytic DB 内部的数据存储结构，并进行数据计算及部分 sql 解析工作。ComputeNode 分为两副本，用户数据保存两份。

- BufferNode：数据缓冲节点，实时写入的数据经过 FrontNode 直接写入 BufferNode，由 BufferNode 负责将写入的数据同步到 pangu 进行持久存储，同时 ComputeNode 也会实时从 BufferNode 同步数据，更新计算节点上的数据。
- DataAgent DataAgent：是 Analytic DB 上一个特殊模块，属于 agentdb 这个数据库，每台机器上都会启动一个 dataAgent 进程，起作用是在数据上线时，将数据从 Pangu 或 MaxCompute 上下载到本地磁盘。

FrontNode/ComputeNode/BufferNode/DataAgent 在飞天上作为 instance 启动，统称 ECU(Elastic Computing Unit)。

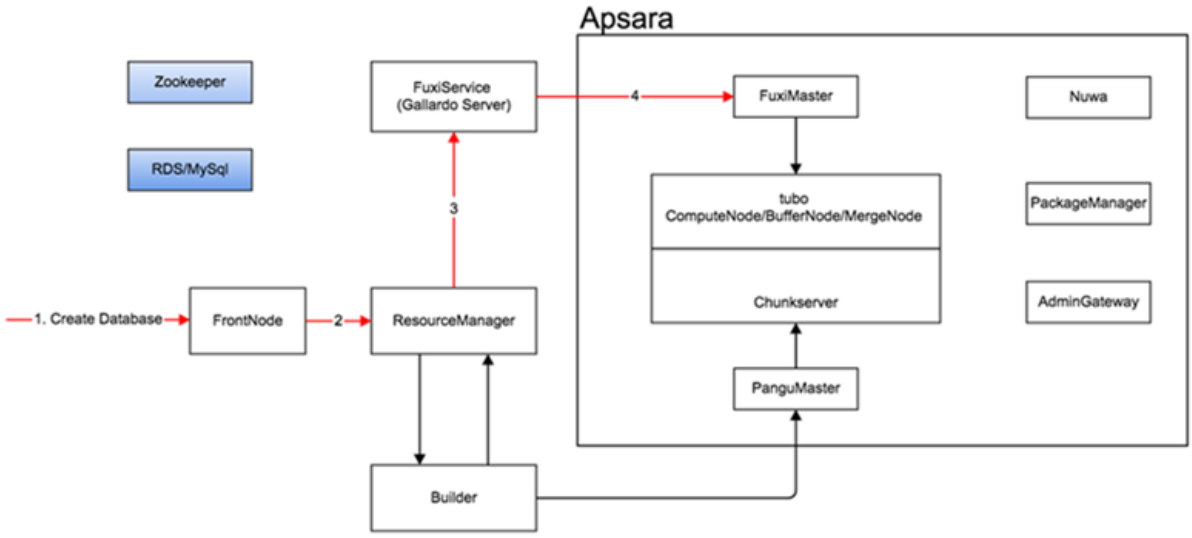
8.4.3.4.1 基础架构及工作原理

Analytic DB集群包含三大组件：飞天，FuxiService和Analytic DB。其中Analytic DB又包括控制节点，计算节点等，几种角色相互配合完成 Analytic DB的所有功能。以下章节会重点介绍每种功能对应的Analytic DB内部的工作原理。

创建DB

创建DB的操作需要使用DMS或连接到SYSDB才能操作，过程如下所示。

图 8-6: 创建DB



1. 用户执行Create database需要传入几个参数：DB 名称，所用的 ECU 类型及个数。
2. SYSDB 的 FrontNode 接到用户的 Create DB 请求后，进行 SQL 解析，将解析后的各参数传给 Analytic DB 的管控节点 ResourceManager 。

3. RM 接到 FN 发送的请求后，根据 ECU 类型，从元数据库中查到此类型所对应的每个模块（FRONTNODE/COMPUTENODE/BUFFERNODE）的资源（CPU/MEM/DISK/NetIO）大小，并将其传给 FuxiService 的 Gallardo Server 模块申请资源，每个模块都叫做 App。
4. FuxiService 接到 RM 请求后，向 FuxiMaster 申请资源，申请成功后，FuxiService 就会到对应的机器上启动相应模块的进程。

下面举例说明如何查看创建DB的进度。

1. 用户创建 db test，占用两个 c1 ecu。
2. 进入 gallardo ui 的 scheduler 页面，可以看到用户申请 DB 的所有信息，包括 App master 的 ip，申请的资源以及已分配到的资源等，如图 *8-7: DB*信息所示。

图 8-7: DB信息

Pool Group	Pool	address	status	Scheduling	Preemptable	RequestMax	WORKER						
							Sessions	RequestedSlots	GrantedSlots	PengdingSlots	ShareSlots	MinSlots	MaxSlo
rmscheduler	new_dmp_5-BUFFERNODE-2		running	FAIR	false	false	1	8	8	0	8	4000	4000
rmscheduler	new_dmp_5-COMPUTENODE-0		running	FAIR	false	false	1	192	192	0	192	4000	4000
rmscheduler	new_dmp_5-FRONTNODE-1		running	FAIR	false	false	1	24	24	0	24	4000	4000

Pool 为 appname，每个 DB 有三个 App，Status 为 App 状态，Worker 一栏中 RequestedSlots 为此 App 申请的资源，GrantedSlots 为已分配的资源，如果 GrantedSlot 和 RequestedSlots 相等，则证明已完成资源申请，FuxiService 正在拉起 Analytic DB 进程。

3. 进入 gallardo UI 的 AmApp 页面，可以看到您申请的所有 App，单击最后一列的 Application Name 进入详情页面，页面中展示了此 App 对应的所有进程信息，包括所在机器，进程 pid，进程启动时间，每个进程所占资源情况等。这些信息对查问题有非常大的作用。

您创建 DB 成功之后，可以得到相应的 jdbc 连接串，在有负载均衡和 DNS 的专有云环境中，返回的应该是一个域名+端口。在没有 DNS 的一体机种，返回是一个 VIP+端口。通过这个JDBC连接串，您即可进行数据库建表和增删改查操作。

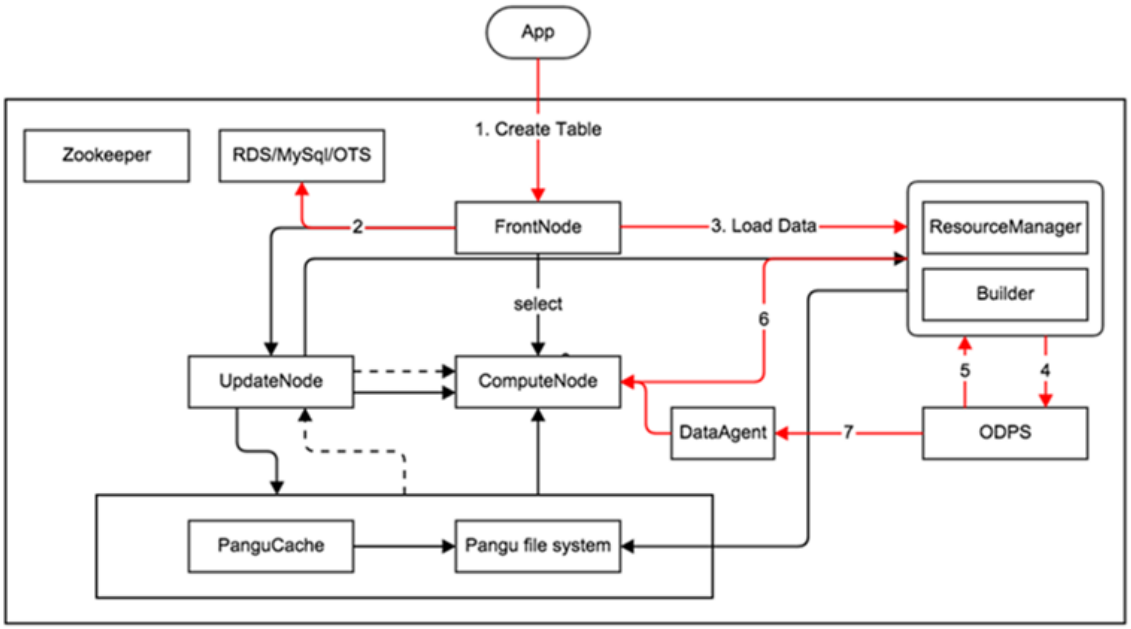
SYSDB 介绍请查阅相关章节，create database 语法请参见：《*Analytic DB* 用户指南》。

创建表

Analytic DB 中，表分为批量导入和实时插入两种类型，在建表时作为参数传入，两种类型的建表流程有部分差异，并且作为分布式数据库，Analytic DB建表流程与传统数据库差异较大，两类表的建表过程介绍如下。

- 批量导入表（batch）

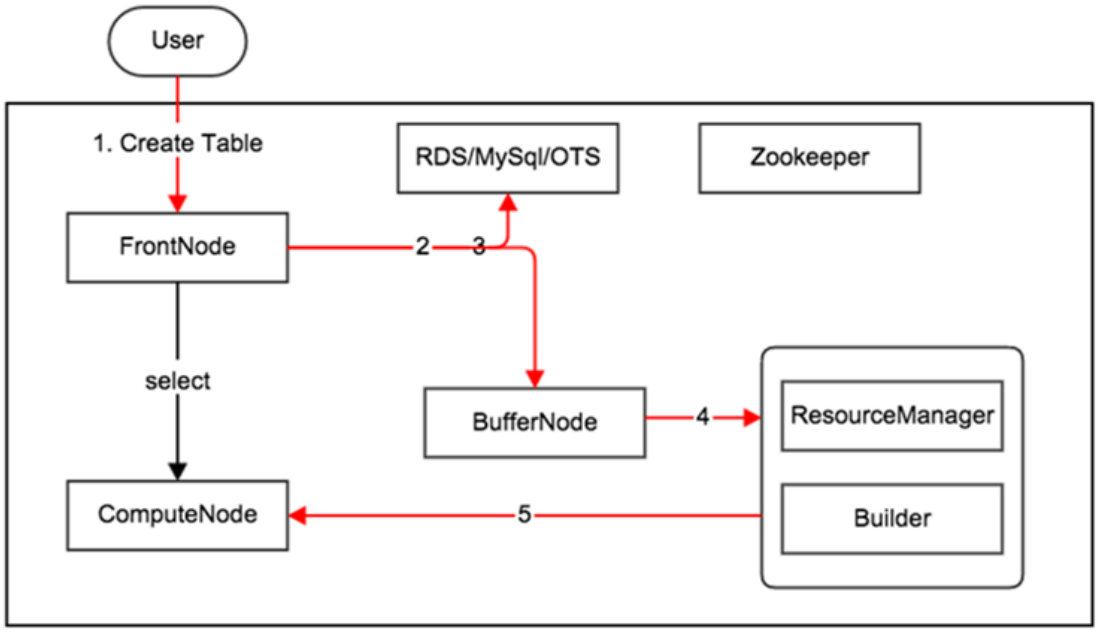
图 8-8: 批量导入表



1. 用户 DB 的 FrontNode 接到用户的建表请求。
2. FrontNode 进行 SQL 解析后，将表的相关元数据，如分区数，表类型，所属表组等信息写入元数据库中。此时，若查询此表会报错 Table not ready，表示此表还没有上线。
3. 用户 DB 发起导入命令（load data），RM 接到数据导入命令之后，根据建表时定义的分区信息，算出每个 ComputeNode 需要存入的表的分区并通知各 ComputeNode。
4. ComputeNode 接到 RM 通知之后，上线相应的数据，并将本节点的数据信息汇报给 FrontNode 此时，数据可查。

- 实时插入表（realtime）

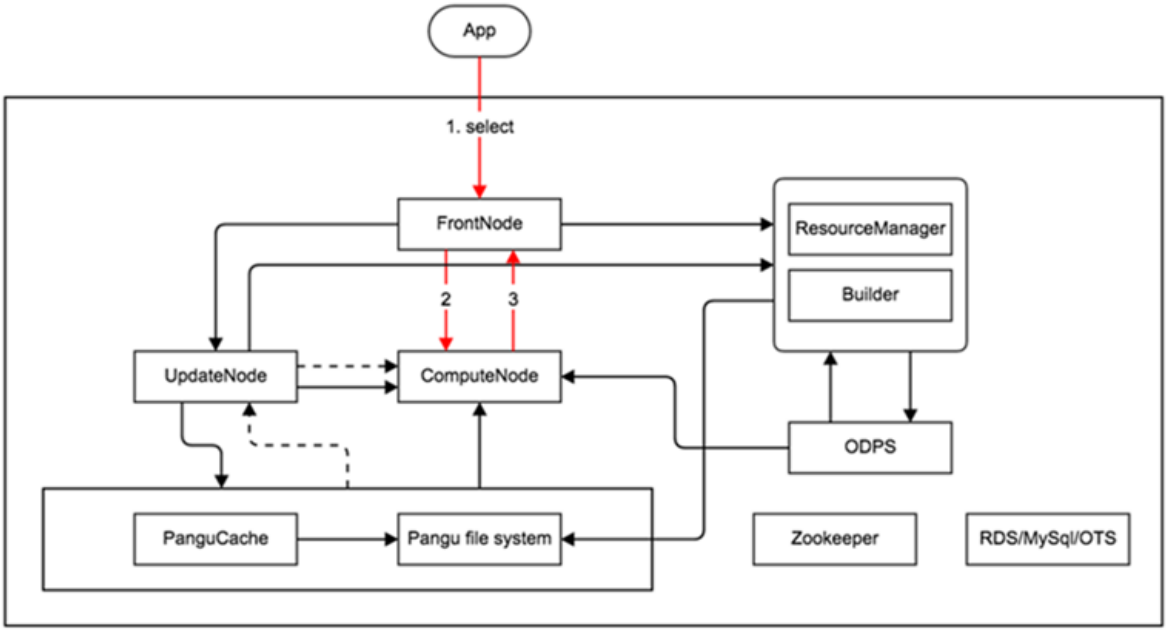
图 8-9: 实时插入表



1. 用户的 FrontNode 接到建表请求。
2. FrontNode 进行 SQL 解析后，将表的相关元数据，如分区数，表类型，所属表组等信息写入元数据库中。此时，若查询此表会报错 Table not ready，表示此表还没有上线。
3. FrontNode 同时将建表请求转发给 BufferNode。
4. BufferNode 接到建表请求后，将请求转发给 RM，RM 根据建表时的分区信息，为各个 ComputeNode 分配分区。
5. ComputeNode 接到 RM 分配的分区信息后，将本节点的分区信息汇报给 FrontNode 此时建表成功，数据可查。

数据查询操作

图 8-10: 数据查询



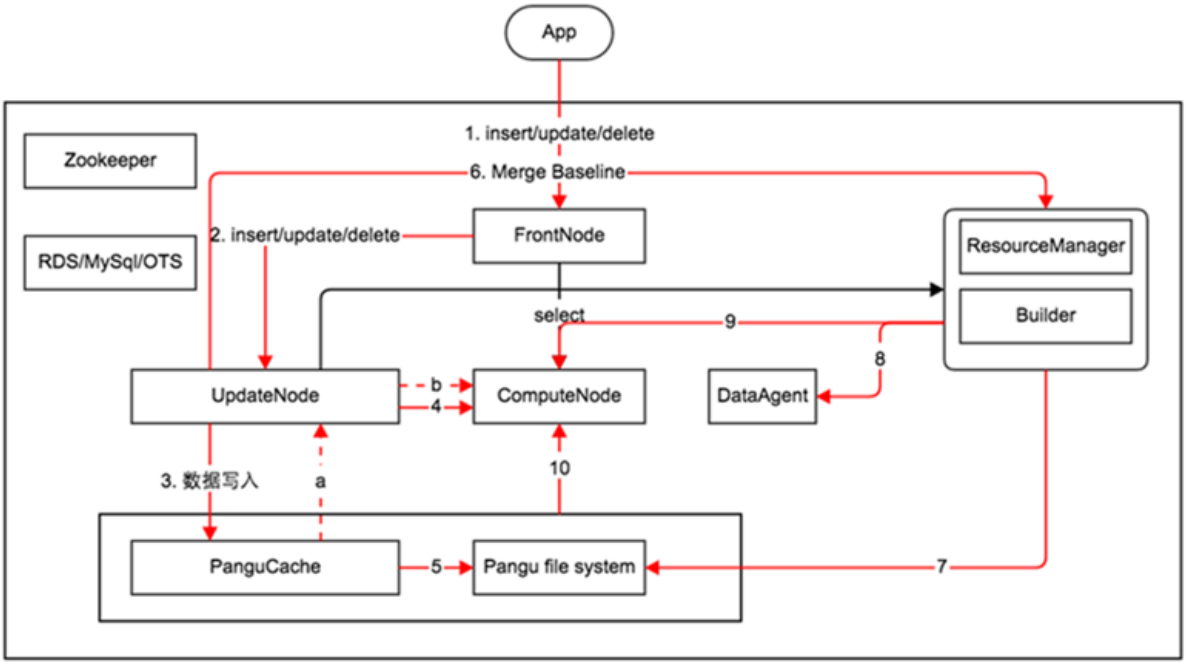
0.8版本中，Analytic DB 中，ComputeNode 分为两副本，每个副本分别保存一份用户数据、保存在 ComputeNode 进程相应的目录下，ComputeNode 每 10s 向 FrontNode 汇报一次数据的分区版本信息。FrontNode 根据 ComputeNode 的汇报分区信息来下发 SQL。select 语句执行流程如下。

1. 用户 DB 对应的 FrontNode 接到用户的SQL，FrontNode 进行 SQL。
2. FrontNode 根据 ComputeNode 汇报的数据分区及版本信息，将 Sql 下发到合适的 ComputeNode 节点。
3. ComputeNode 接到 FrontNode 下发的 SQL，对本地数据进行计算并将计算结果返回给 FrontNode 4. FrontNode 对所有 ComputeNode 返回的数据进行汇总，并将最终结果返回给用户。

数据更改操作

数据更改包括增删改三种操作，Analytic DB 中，这三种操作的流程较为复杂，详情如下。

图 8-11: 数据更改

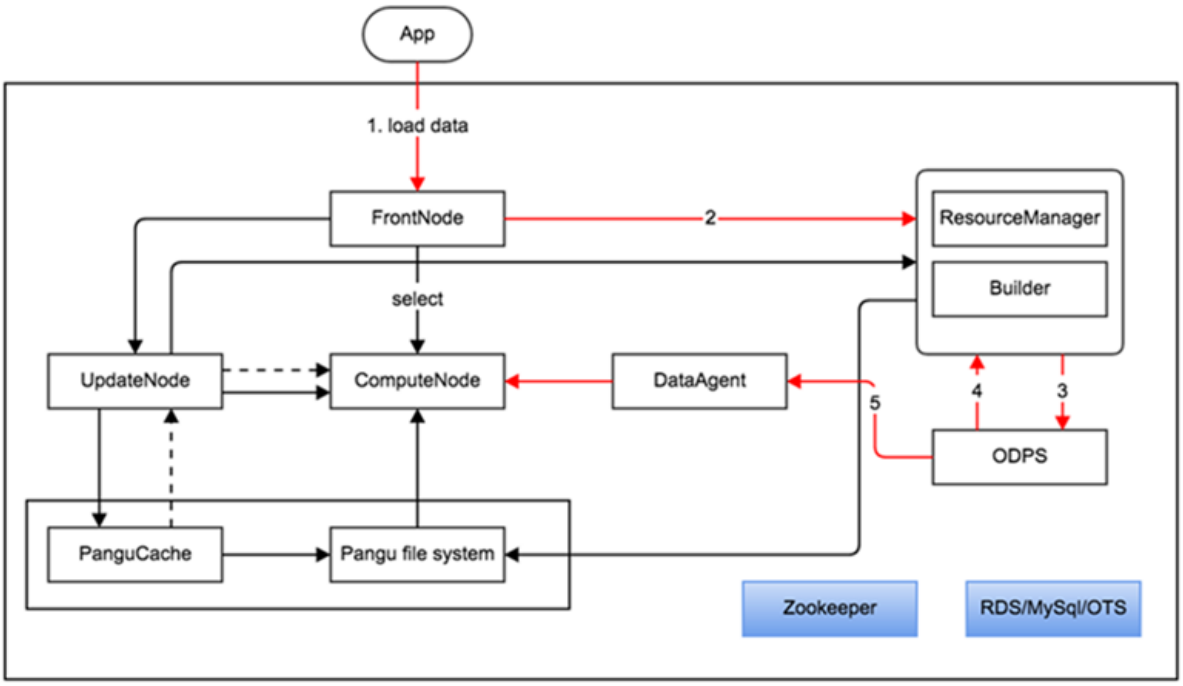


1. 用户发送 insert/update/delete 命令到 FrontNode。
2. 用户 DB 的 FrontNode 接到 SQL 之后进行 SQL 解析，并将 SQL 下发给用户 DB 对应的 UpdateNode（UN）。
3. UpdateNode 拿到数据之后，将数据写入 PanguCache。
4. 同时，ComputeNode 也会定期从 UpdateNode 同步数据，并对本地数据进行版本更新。为了保证查询性能，ComputeNode 上的数据，都保存在 SSD 上，Analytic DB 的 ComputeNode 为2副本，共保存两份用户数据。
5. Pangu 会从 Pangu Cache 同步数据，为节省成本，Pangu 上持久存储的数据都保存在 SATA 盘上，目前保存3副本。
6. 每天在特定的时间段（可配），UpdateNode 会对所有实时表发起 Merge Baseline 操作。Merge Baseline 会将 pangu 上当天的数据构建索引，并将结构化之后的数据存储到 ComputeNode 上。Merge Baseline 会发起构建索引和上线过程。
7. RM 接到构建索引命令后，会通知 build 在 pangu 上开始构建索引，build 开始构建索引任务并将任务状态同步给 RM。
8. RM 接到数据 build 任务完成之后，会通知 DataAgent 从盘古上下载相应数据到指定目录。
9. DataAgent 数据下载完成后，RM 会通知 ComputeNode 开始上线数据，上线完成之后，ComputeNode 的本地磁盘上会保存完成的有强索引的数据，以此保证实时数据的查询速度。

数据导入

在存在 MaxCompute（原ODPS）的专有云或一体机中，您可以通过 load data 命令直接把 MaxCompute 上的数据批量导入 Analytic DB 中，基本的导入流程如下。

图 8-12: 数据导入流程



1. 用户发起导入命令（load data），load data 命令需有MaxCompute上源表的地址，Analytic DB 上目的表的表名及分区信息等，用户DB的FrontNode接到导入命令后，进行sql 解析，并把请求转发给ResourceManager。
2. ResourceManager 根据 MaxCompute源表信息通知builder模块，对用户的数据进行build，并将build结果保存在Analytic DB对应的MaxCompute project中。
3. Builder模块会轮询构建索引的mr任务的状态信息，一旦完成，就会把信息同步给ResourceManager，相关索引数据及用户原始数据都会保存在Analytic DB的MaxCompute project中。
4. ResourceManager接到构建索引完成的指令之后，会通知相应的ComputeNode机器上的dataAgent，将数据从 MaxCompute上下载到本地磁盘的特定目录下，下载完成后，ResourceManager会通知相应的ComputeNode进程进行上线操作。 ComputeNode数据上线的过程就是遍历本地数据的过程，ComputeNode得到本地最新数据后，将信息汇报给MergeNode，之后用户即可查到最新数据。在ComputeNode遍历本地文件的过程中，此ComputeNode会

暂停服务，其上的数据会不可查。Analytic DB本地数据保存在ComputeNode对应的目录下，ComputeNode分为两副本，每次同时只会有一副本在上线，以保证用户查询。

load data命令的用法请参见《*Analytic DB* 用户指南》 。

8.4.3.4.2 目录结构

Analytic DB 所有模块都是 java 程序，均使用 virgo 容器启动，所有模块 virgo 容器的内部结构大致相同，主要路径及文件介绍如下。

garuda -- 所有模块 virgo 容器的根目录都是 garuda。

|-- bin -- 进程启动脚本所在目录。

| |-- garuda.pid -- 进程启动后，process id 的存放文件。

| `-- garuda.sh -- 进程启动脚本。

|-- logs

| |-- garuda.log -- 执行 garuda.sh 后返回结果的输出文件，常用来定位进程无法启动的问题。

| |-- gc.log -- 进程 gc 日志。

`-- virgo -- virgo 主目录。

|-- bin -- virgo 容器启动脚本的目录。

| |-- startup.sh -- virgo 容器的主要启动脚本。

|-- config

| |-- serviceability.xml -- ads 程序日志格式文件，描述每个模块所有日志的打印格式，轮转规则等信息，每个模块都不相同。

|-- etc -- ADS 各模块启动所需的配置文件。

| |-- config.ini -- 加密文件，保存了各个模块启动所需的所有 id 和 key，具体内容会在下文介绍。

| |-- config.key -- 私钥，程序反解 config.ini 文件所待的私钥。

| `-- zookeeper.properties -- 保存 zk 的链接地址。

|-- lib -- virgo 容器本身的 lib 库。

|-- pickup -- 保存了 ADS 各模块的 jar 包，virgo 容器启动时会加载这个目录下的 jar 包，程序升级即是要替换这个目录下的 jar 包。

| `-- com.taobao.garuda.resourcemanager.resourcemanager-server-0.8.5.2.jar -- ADS 程序 jar 包。

|-- virgologs -- ADS 模块日志的保存路径，所有模块的日志都保存在这个路径下。

`-- work

| |-- tmp -- 在 COMPUTENODE 中，是用户数据保存目录，保存此节点上加载的数据。

 **说明**：

ResourceManager 和 Builder 目录结构 ResourceManager 和 Builder 是在机器上作为独立进程手动启动，因此目录结构非常固定。RM 和 builder 根目录是*/home/admin/garuda*，各目录作用如上文所述，常用目录如下。

- 启动进程 --/home/admin/garuda/bin/garuda.sh stop
- 停止进程 --/home/admin/garuda/bin/garuda.sh start
- 查看日志 -- /home/admin/garuda/virgo/virgologs/

飞天拉起模块的路径和目录结构

FRONTNODE/COMPUTENODE/BUFFERNODE 三个 ADS 模块由飞天拉起，启动时待的 virgo 容器，启动脚本和配置文件都保存在 pangu 上。进程第一次启动时，飞天会从 pangu 上固定的路径来获取这些文件。

- ComputeNode/FrontNode/BufferNode 启动所需 virgo 容器在 pangu 上的路径如下。*pangu://localcluster/garuda/repository-${clustername}/frontnode/${version}/package/garuda.tar pangu://localcluster/garuda/repository-${clustername}/computenode/${version}/package/garuda.tar pangu://localcluster/garuda/repository-${clustername}/buffernode/${version}/package/garuda.tar*
- ${version} 为每个模块对应 jar 包的版本信息 garuda.tar 即为每个模块的 virgo 容器，每个模块的 garuda.tar 解压后都包含一些 virgo 容器的基本路径。

  garuda/etc -- 保存配置文件的路径

  garuda/lib-- 保存程序jar包

  garuda/bin -- 保存启动脚本

  garuda/config -- 保存各个模块的日志配置文件 serviceability.xml

  注意：每个模块的 garuda.tar 的内容都不相同。

- ComputeNode/FrontNode/BufferNode 启动所需配置文件在 pangu 上的路径。*pangu://localcluster/garuda/repository-${clustername}/config/${version}/config.tarpangu://localcluster/garuda/repository-${clustername}/config/${version}/config.key*

  ·${version}为每个配置文件的版本信息

  如上文所述，Analytic DB 各模块启动时，需要读取 virgo 容器中 etc 目录下config.ini 配置文件解密之后的内容，而进程初次启动时，就会从 pangu 上的 config 目录下，拉取对应版本的 config.tar 和私钥 config.key，并将其解压，放入 virgo 容器的对应目录。 config.tar 解压之后是 etc 目录，内部包括此版本对应的 config.ini 文件。config.key 是用来解压的私钥。
- ComputeNode/BufferNode/FrontNode 启动脚本在 pangu 上的路径。*pangu://localcluster/garuda/repository-${clustername}/dbmanager/${version}/script/install.sh*${version} 为当前系统中 ResourceManager 的版本，此版本一定要和 ResourceManager 的版本一致 install.sh 是所有飞天拉起进程启动时待的启动脚本，负责文件解压，进程启动等。

8.4.3.4.3 查找和阅读日志

日志记录了系统运行状况，对日常巡检和故障诊断都非常重要。Analytic DB 各个模块都记录的详细的运行日志。 ResourceManager&Builder ResourceManager/Builder 进程由人工启动，日志目录固定，为 RM/Builder 机器上的*/home/admin/garuda/virgo/virgologs*目录。

模块	日志名称	关键字
ResourceManager&Builder	log.log	进程启动日志，用来监测进程启动是否正常
ResourceManager	resourcemanager.log	业务日志，记录资源申请及释放，系统升级，数据上下线等信息
Builder	load.log	业务日志，记录各build任务状态等

ComputeNode&FrontNode&BufferNode 这三个模块由飞天拉起，日志目录不固定，有以下方式可以查看。

- 通过 Gallardo UI Gallardo UI 上记录了所有进程所在的主机 IP、PID、日志路径和资源使用情况等，可以直接在 Gallardo UI 上查看各进程日志。

  1. 登录 Gallardo UI。在专有云和一体机种，Gallardo UI 的地址为：*http://${agIP}:8315/index* ` *agIP*`为此飞天集群 AdminGateway 机器的 IP 地址。
  2. 单击AmApp，单击所要查看日志的进程的 Aplication Name，例如，查看 testdb 的 BufferNode 的日志，就单击testdb_${dbid}-BUFFERNODE-2，进入进程详情页面。

3. 进入进程详情页面之后，单击某个进程的 logs 列，即可进入此进程日志文件页面。
4. 单击需要查看的日志文件，即可看到整个日志。

> **说明**：
> 在查看日志时，要注意日志文件开始打印的时间，Analytic DB 的日志按时间和大小两个维度轮转，有可能一天会有多个日志。

- 登录机器进行查看。每个飞天拉起的进程启动时，都会将进程自身信息注册到zk上，包括进程所在机器和日志路径。

  例如：testdb 的 FrontNode 进程启动时，会将所在主机信息及日志路径注册到 /global/mnmg/db/testdb/service/${ip_port} 中，在 console 上找到此节点，单击之后右侧就会显示此进程启动时，注册上的详细信息，host 即为机器名，workDir 即为日志路径，登陆此机器，进入日志路径，即可看到所有日志。

各模块主要日志及作用如下。

表 8-2: 各模块日志及作用

模块	日志	作用
FN/CN/UN/dataAgent	log.log	记录进程启动信息
FN	access.log	记录FrontNode接收到的所有SQL及总执行时间
FN	profile.log	记录FrontNode接收到的每条执行成功的SQL的每个阶段的执行时间
FN	exception.log	记录所有FN的异常信息
FN	analysis.log	记录所有SQL的详细执行信息，包括所有下发到CN上的分区以及各CN的返回信息，对排查问题非常重要，其中有各个SQL的pid，会透传到各个CN上，根据此PID可以找到有关此SQL的所有阶段的执行信息
FN	dns.log	新建DB时，此日志会记录dns注册信息
FN	slb.log	记录此FN instance注册负载均衡的详细信息
ComputeNode	engine.log	各SQL执行情况及CN上线情况
ComputeNode	exception.log	记录CN所有异常信息
BufferNode	httpaccess.log	记录BufferNode接到的所有请求，包括源IP等

模块	日志	作用
BufferNode	queuestorage.log	记录本台BufferNode存储层日志
BufferNode	updatestatus.log	记录本个BufferNode的状态信息，如qps，10s内读写次数，字节数，慢查询，慢写入等等
dataAgent	engine.log	记录dataAgent所有操作，包括下载数据等

8.5 系统元数据库和SYSDB介绍

8.5.1 基本信息元数据表

作为分析型数据库，分析型数据库的元数据中包含了描述用户数据的信息，包括 DB 名，表明列名及各种属性等，同时，Analytic DB 数据库中也包含了程序内部任务执行的状态，过程等。目前共有 56 张元数据表，本章内容会详细介绍其中常用表的结构及内容。Analytic DB 的元数据库一般采用 MySql。

schemata/tables/columns/indexes/table_groups 保存了用户在 Analytic DB 中所有 DB 及表的基础信息。

8.5.2 权限相关元数据表

- users ：保存了 Analytic DB 所有用户的 aliyun 账号以及对应 DB 的连接域名 domain_url。
- administrator：数据库管理员权限，在此表中的权限可直接连接 sysdb 并且查看所有元数据表，包括 mysql 中的元数据表，以及 instance_profile 和 query_profile 两张数据元仓的表。
- privilege_schema：DB 级别用户授权，在 Analytic DB 中，可以将某个 DB 对特定用户授权。则该用户拥有对此 DB 中所有表的数据的权限。
- privilege_table：表级别的用户授权，在 Analytic DB 中，可以将 DB 中的某张表对特定用户授权。则该用户只能看在这个 DB 中特定表的数据。
- privilege_column：列级别用户授权，可以将 DB 中某张表的某一列或几列对特定用户授权，则该用户只能看到这几列的数据。

8.5.3 数据导入相关的元数据表

分析型数据库中，数据导入过程，包括生成导入任务，构建索引，数据上线三大过程。与此过程相关的元数据表中包含了数据导入过程各个阶段的所有任务以及任务的状态等信息，对运维排查问题方面有非常重要的作用，本章节重点介绍与数据导入相关的所有元数据表，列表如下。

- table_data_loads ：记录用户发送导入任务时的数据源、账号、请求源 IP、表分区和数据 version 等基础信息，用于记录用户导入历史。 Analytic DB 数据有两个数据源，实时数据的数据源是 pangu，而批量导入数据的数据源是 MaxCompute（原ODPS），用户可根据实际情况选择是否使用 MaxCompute。 表分区分为两种情况，如果是一级分区， table_partition 列只存储分区列名，如果是二级分区表，则 table_partition 列格式为：/一级分区列=/二级分区列=二级分区 data version。
- job_info ：Analytic DB 中每次上线操作都对应一个 job，此表保存所有上线操作的 job 的汇总信息，如申请时间、申请状态、数据操作类型 PUT_DATA（上线）、REMOVE_DATA（下线）和申请状态等。
- current_job ：保存当天所有导入的汇总信息，包括所属 DB、表、导入分区信息、导入开始结束时间以及任务状态等，其中 JOB_ID 独立标示一个上线任务，通这个 JOB_ID 可以在不同元数据表中查到这个上线任务的各个阶段的状态。
- history_job：保存历史所有导入任务的汇总信息，每天24点归档current_job中的所有已完成任务信息，包括 SUCCEEDED 和 FAIL 状态的任务。

  该表中的任务包括了每个任务的任务详情，包括三个阶段的起始时间及任务状态等，为分析每天数据导入任务的执行时间及成功率等提供了非常详细的信息。
- current_task： Analytic DB 中的每个导入任务都称为一个 job，每个 job 又分为3个阶段即3个 task 来完成，而此表就是生成 job 时插入的。

  ResourceManager： 接到导入命令，生成一个导入任务时，会在此表中插入3条数据，分别对应导入任务的三个阶段。

  对于这个导入任务会生成一个唯一的任务id即 JOB_ID，对每个 task 又会以 JOB_ID 为基础生成一个唯一的 TASK_ID 来标识。此表中保存了一个任务的每个阶段的起始时间，任务状态，输出结果等信息。此表只保存当天运行任务的信息，默认每隔 24 小时清理一次已完成的任务列表（包括状态为 SUCCEEDED 和 FAIL 的任务）。
- build_current_task ：Analytic DB 中导入任务的第二个阶段构建索引在 Analytic DB 中通常称为数据 build，数据 build 阶段中，会对导入任务中的表在 MaxCompute 上起一个 MR 任务进行计算，此表即保存了 build 过程中的详细信息，包括表名，开始结束时间，MaxCompute 上 Split/Merge 任务的 logview 地址等。此表中只保存当天数据，所有历史数据都会被归档到 build_history_task 中。
- build_history_task： build_current_task 的历史数据归档表，保存所有历史导入任务数据构建（build）阶段的详细信息。

- table_data：目前计算节点上可查的每个表的数据版本（data version）。
- table_data_request：记录数据导入的第三部分--数据上线请求的元数据表，此表中记录了每次上线请求的开始时间 request id，任务状态等信息。此表中请求类型（request_type）分为两类：PUT DATA 和 DELETE DATA。PUT DATA 是数据上线，DELETE DATA 是数据下线。
- table_data_request_task ：每个上线任务又以机器为单位，被拆分成多个上线 task，ResourceManager 会计算出每台 COMPUTENODE 上应该上线哪个分区，并将计算结果保存到此表中的 shards 列。 ResourceManager 会根据此表中的信息通知所有 COMPUTENODE 分别对相应分区做相应的操作。
- table_partition_index：记录二级分区表的每个二级分区的数据版本，partition_version 列记录了二级分区表的每个二级分区号对应的二级分区的版本号，格式为二级分区号=二级分区数据版本，partition_version 中保存的二级分区是用户指定的保存时间内的二级分区，如果超过保存时间，则会被删掉。

8.5.4 资源申请相关元的数据表

分析型数据库的COMPUTENODE，FRONTNODE和BUFFERNODE三个模块是由飞天拉起的，因此这三个模块中进程的创建，释放都需通过FuxiService向Fuxi申请。一下几张元数据表就记录了关于资源申请，释放相关的信息。

- resource_type：记录Analytic DB不同资源类型的每种进程的具体资源占用信息。Analytic DB中可以根据不同情况配置多种资源类型，每种资源类型中COMPUTENODE/FRONTNODE/BUFFERNODE所占用的CPU/MEM/DISK/NETWORK资源都不同。不同业务可根据自身需要申请不同的类型的worker来达到节约成本的作用，这点在ADS公共云体现尤为明显。在专有云中，客户也可以根据自身需求调整资源类型的出厂设置，以达到最好的使用效果。
- cpu：列为资源类型所占CPU核数，Analytic DB中CPU可以超卖，超卖时各worker可以按时间片共享CPU。mem/disk/network_io分别表示每种worker可占用的内存，磁盘以及网络io大小，目前disk_iops磁盘读写io尚未做隔离，因此此参数目前无效。Analytic DB会根据这个表中的配置，向Fuxi分配资源大小，由Fuxi来进行资源隔离。
- resource_group：记录每个 DB 中所有已申请的 worker 的包版本和配置版本信息。
- resource_node：记录每个 DB 中所有已申请 worker 的类型，worker_id、 external_worker_id 、资源组、所属副本以及资源占用情况等信息。其中 worker_id 是只在 DB 内部 worker 的 id ，例如一个 DB 中有10个 worker，那么 worker_id就是从1到10。external_worker_id 是外部 worker ID，由 DB、资源组等信息拼成。

- resource_request：记录了每个资源申请的基本信息，包括表名，系统分配的上线 request id，request 类型，上线状态等。

  其中 request_type 有两种。

  - ALLOCATE：申请资源启动 worker。
  - REMOVE：停止 worker 释放资源。
- resource_request_task：每个资源申请都会以 worker 为粒度，被拆分成多个资源申请 task，申请下来的所有 worker 会被 ResourceManager 进行标记，如分配资源组，分配 worker_id，external_worker_id 等。为之后的数据上线做准备。
- db_resource：每个DB的资源类型及worker数量，同时保存了DB的用户名和id用作审计用途。
- db_resource_config：DB特殊配置表。

8.5.5 实时表相关的元数据表

- realtime_baseline：保存实时表每天 merge baseline 的信息，包括表名，baseline 的 job id，baseline 在 pangu 上的数据路径，baseline status 等。其中 baseline 状态有以下几种。

  - -1： ERROR，merge baseline 失败
  - 0： INIT，初始化
  - 1：MERGEING
  - 2： BARRIERING
  - 3： BUILDING
  - 4： FINISH

  0，1，2，3 为 merge baseline 的中间状态，其中 building 持续时间最长，一般用来确定是否有表在 merge baseline，如果状态是-1，buffernode 会一直重试，直到状态变成 4 或被人为干预。

  目前此表会对每张表保留30天的 merge baseline 的历史数据。
- realtime_sync_version：保存每个实时表每个分区当前的正在写的数据版本，可读的数据版本以及写 qps，每 10s 更新一次，可用来统计每张表的写入 qps 信息。

  说明：

  此表实时更新，并不保存历史数据。
- realtime_tablestatuses：保存了每张表 merge baseline 的状态信息，用作在某些灾难场景下恢复 zk 上每个表 merge baseline 的状态信息。

8.5.6 其他元数据表

- slb_instance：在开启了负载均衡配置的专有云或一体机环境中（Zookeeper 中/global/config /master 下 withSLBEnv 设置为 true），在部署时会调用负载均衡的接口来申请一个或多个指定类型的负载均衡实例（VIP），以后在创建 sysdb 和其他用户 DB 的过程中，FRONTNODE 启动的时候会读此表的负载均衡实例，为拉起的 DB 分配一个 VIP+端口并写入到对应 DB 的 schemata 元数据表中。
- instance_rs_pool：记录 instance 和机器实际端口的的对应情况。ADS 的每个 DB 都会拥有多个 instance，每个 instance 对应启动在某个 IP 和 port 的进程，此表记录了 DB 和 IP+PORT 的对应关系。
- partitions：存储所有表的分区信息，包括表明，分区号，此分区的数据版本以及数据大小，通常用来计算每个表的大小。 注意：此表中包含了每个的所有的历史数据版本，计算表数据量时，待判断目前线上 data_version。

8.5.7 sysdb

Analytic DB（原 ADS）在元数据库的基础上，设计了系统数据库 sysdb，sysdb 是在 Analytic DB 上直接创建的数据库，可以像用户数据库一样进行读写操作，sysdb 本身有两张表：instance_profile 和 query_profile，与其他用户数据不同的是，通过 sysdb，可以直接查询到 mysql 上的所有元数据表进行读写操作，是 Analytic DB 提供给系统管理员的系统管理及运维的总入口。通过 sysdb 可以进行以下操作。

1. 新建数据库在 Analytic DB 专有云和一体机中，用户一般通过 DMS 界面进行建库建表操作，同时，系统管理员也可以通过 mysql 客户端直连 sysdb 或 admin 库创建 DB。
2. 查询所有的元数据通过sysdb，可以查到所有元数据库中的信息，注意，查询时待在mysql链接串中加上 -Dsysdb，并且在查询中带上sysdb库名来查表，同时在 sysdb 中，目前 show tables 无法显示出元数据表表名。
3. 查询系统性能元数据表 instance_profile 和 query_profile。
   - instance_profile instance：资源使用状态表，记录了每个 instance 的运行时资源使用信息，每个 instance 在运行时都会一分钟更新一次此表中的数据，内容包括每个 instance 申请的 CPU，内存，硬盘，网络 io 以及磁盘 io，同时还记录了每个 instance 已经使用的 CPU，内存，硬盘等资源信息，对运维来说非常重要。
   - query_profile：记录每个 query 的执行时间，返回条数等性能信息，集群查询性能分析有非常重要的作用。

说明：

两张表在 mysql 元数据库中查不到，仅存在 sysdb 中。

连接 SYSDB 需要有专门的 accessKey 和 accessSecret，普通用户没有权限访问，只有系统管理员才有权限访问。

8.5.8 information_schema

information_schema 是 Analytic DB 提供给用户，供用户查询的元数据表。

用户使用自己的账号登录 Analytic DB 数据库后，可以通过 information_schema 查询本 DB 的如下元数据信息： schemata，tables，tablegroups，columns，partitions，indexes，statistics，servers，currentJob，historyjob，currentinstances，resourcerequest，keycolumnusage。

正常情况下，用户没有权限访问 Analytic DB 的元数据库和 SYSDB，information_schema 是用户查询本 DB 相关元数据的唯一接口。

常用表如以下所示。

- information_schema.schemata：查询数据库配置。
- information_schema.table_group：查询数据库表组信息。
- information_schema.tables：查询用户 DB 表信息。
- information_schema.current_job：查询用户 DB 数据导入任务。
- information_schema.current_task：查询用户 DB 数据导入任务。

8.6 部署架构

8.6.1 部署角色

整个 Analytic DB 的集群包括两种节点类型：控制节点（master）和计算节点（slave）。

- master 机器机型选用 W2
- slave 机器机型选用S10-3S

master 节点的数量及主备关系如下所示。

模块	角色	机型	机器数量	主备方式	节点类型	装机模板
apsara	盘古master	W2	3台	主备	master	bigdata
apsara	伏羲master	W2	2台	冷备	master	bigdata

模块	角色	机型	机器数量	主备方式	节点类型	装机模板
apsara	nvwa	W2	3台	主备	master	bigdata
apsara	Package Manager	W2	2台	主备	master	bigdata
apsara	Admin Gateway	W2	1台	单点	master	bigdata
apsara	slave -- tubo/ chuncserver	S10-3S	>3台	分布式	slave	ads_cloud_server
FuxiService	Gallardo Server	W2	2台	主备	master	bigdata
FuxiService	Gallardo UI	W2	2台	主备	master	bigdata
FuxiService	Gallardo RMUI	W2	2台	主备	master	bigdata
ADS	ResourceManager	W2	2台	主备	master	bigdata
ADS	Builder	W2	2台	主备	master	bigdata
ADS	ComputeNode	S10-3S	>3台	双副本	master	bigdata
ADS	FrontNode	S10-3S	>3台	分布式	slave	ads_cloud_server
ADS	BufferNode	S10-3S	>3台	分布式	slave	ads_cloud_server

大部分专有云和一体机中，由于集群规模较小，master 节点压力不大，所有的 master 节点都是混布的（即多个角色部署在同一个物理上）。

说明：
一体机种，FuxiService 的三个模块都是单点，只部署在 AG 机器上。

8.6.2 部署框架

8.6.2.1 前置依赖

在专有云中，Analytic DB（原ADS）依赖的外部系统较多，包括以下几种。

- RDS -- 为 Analytic DB 及 FuxiService 提供元数据库。
- 负载均衡 -- 为 Analytic DB 提供 VIP 实例。
- DNS -- 为 Analytic DB 提供 DNS 域名服务。
- MaxCompute -- 一般专有云中，用户都需用到批量导入功能，因此需要 MaxCompute。
- aliyun 中控系统 -- 专有云中整个阿里云平台的账号管理体系，包括 UMM，瑶池和 AAS。

- Admin Gatway 上的 MySql 数据库 -- 为 Analytic DB 及 FuxiService提供元数据库，完成飞天安装 之后即可使用。
- 负载均衡 --单独部署。

其他依赖

- Analytic DB 所部署环境的所有机器，必须有 ntp 同步服务，飞天正常工作强依赖各个组件间的时间戳来同步任务状态，如果某台机器出现严重时间漂移（>1min），则不能正常工作。
- Analytic DB 所有机器必须配置 DNS 路由信息，即配置 IP 到机器名及机器名到 IP 之间的映射关系，飞天工作强依赖机器名。

在正式开始部署AD之前，必须保证所有的前置依赖都完成部署并处于正常工作状态。

8.6.3 分析型数据库本体（ADS）

ADS 部署的所有模块如下所示。

- zk_init：初始化 ZooKeeper，ADS 所有模块启动都会以来 ZooKeeper 上以来的一些基本信息，如元数据库地址，pangu 版本等，在部署其他角色之前，必须先初始化 ZK，初始化 ZK 的配置文件存放在*bigdata/products/ads/modules/zk_init/conf/zkconf.xml.tpl* 中。
- adsconsole ADS：运维平台，用来展示 ZK 上的配置信息及导入任务状态。
- config_init：生成 config.ini 加密文件。config.ini 中保存所有 ADS 跟其他系统交互的用户名和密码信息，所有模块启动都要加载这个文件。
- resourcemanager/builder： ZK 初始化和 config.ini 生成之后，即可启动 RM和 Builder，RM 或 Builder 启动之后，会将在 MySql 元数据库中创建所有元数据表。
- meta_init：初始化元数据信息。RM 正常启动之后，RM 中的 meta engine 会在元数据库中生成所有的元数据表，其中有一些表待在启动其他模块之前进行初始化，如 resource_type，resource_group，users 等。
- mergenode/localnode/updatenode：三个模块部署是向 pangu 相应的目录下放各个模块的 garuda.tar，install.sh 及 config.tar 文件。此时并没有任何进程启动。
- ads_tools：在依赖 ODPS 的专有云或专有域中，待向 ODPS 的 public project 中上传两个资源文件： odpssk-unsafe-0.1-SNAPSHOT-95226886eccd.jar， odpssk-unsafe-0.1-SNAPSHOT-95226886eccd.jar.policy。 这两个文件如果上传不成功，则会导致所有批量导入表 build 不成功。在不依赖ODPS的环境中，此步骤可跳过。
- slb_config：在曙光中，此步骤是通过负载均衡的API申请负载均衡实例，如果执行成功，则元数据库中slb_instance表即会有记录，保存申请的负载均衡实例地址（VIP）及起始端口。

- start_sysdb：在slb_config执行成功之后，才可启动 SYSDB，在专有云和一体机中，启动 SYSDB 时，会调用 ResourceManager 接口，申请2个 C8 实例，如果集群规模较小，可将实例类型改成 C1。 SYSDB 创建成功之后，ADS 部署就算完成，即可进行创建 DB 等其他冒烟操作。

9 大数据应用加速器

9.1 前言

随着近五年互联网和大数据技术的蓬勃发展，各类数据产品应运而生，从阿里自身大数据的应用发展来看可以看到几方面的挑战：

- 一方面为了应对数据量高速的增长，衍生出各类的分布式数据计算与存储技术解决各类应用场景下的难题，而非传统 IT 架构当中只需要单一数据库就可以支撑整个企业的数据分析报表问题；各类数据的积累如何进行有效的整合与管理，各个业务库的数据之间如何打通在多个计算存储资源上合理的分布管理也成为一大难题；
- 另一方面，大数据在各个行业当中的应用，如数字广告、互联网金融、电子商务、在线风控等场景当中，一个数据应用需要囊括报表分析、行为预测、实时监控、信用评分、个性化推荐、文本挖掘、时空数据等各类大数据技术方法的综合运用，而不仅仅是做企业经营的报表统计；
- 并且，当下对运用数据的用户也不只是局限在专业的数据分析师、数据仓库工程师，更多的是能够让非技术背景的业务人员能够以他能够理解的方式灵活的探查数据。

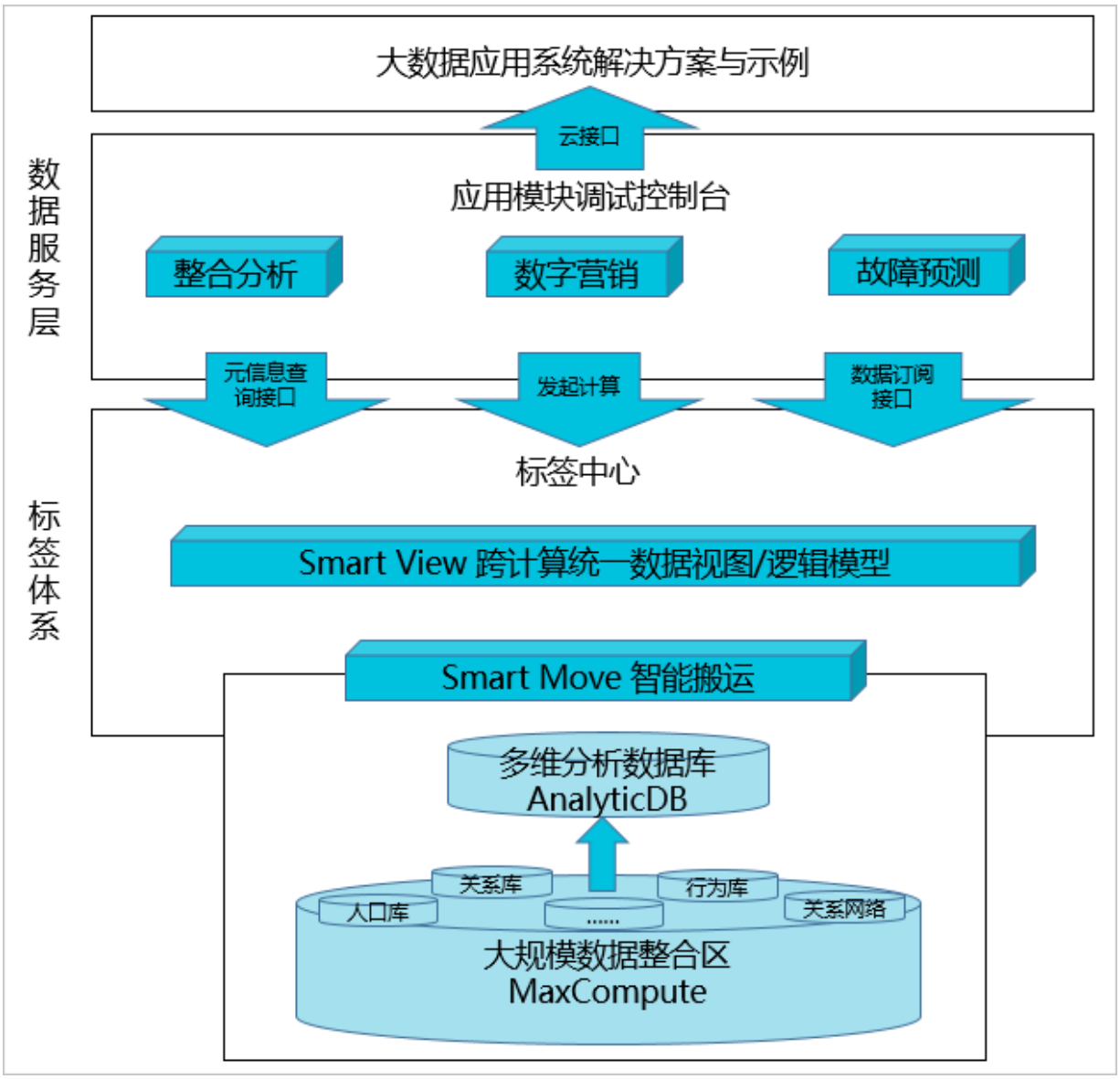
在这三方面之下，如果想要运用好大数据就对企业的 IT 架构、技术人员综合能力的要求提出了更高的挑战，既要能了解各个专业分布式计算和存储资源的特性，又要求能将这些资源针对数据分析、算法服务等多种应用场景进行合理的架构，还要能够针对业务人员使用数据的场景足够了解并告诉相应，制作出面向业务的数据产品。

阿里云 DTBoost 数据加速器产品从大数据应用落地点出发，提供了一套大数据应用开发套件，能够帮助开发者从业务需求的角度有效的整合阿里云各个大数据产品，大大降低搭建大数据应用系统当中绝大部分的系统工程工作，在相应行业应用解决方案的结合下，能够让不是很熟悉大数据应用系统开发的程序员也能够快速为企业搭建大数据应用，从而实现大数据价值的快速落地。

9.2 产品概述

DTBoostv2.0产品组件如图 *9-1: DTBoost v2.0产品组件*所示。

图 9-1: DTBoost v2.0产品组件



概括来讲，DTBoost 是以标签中心为基础，建立跨多个云计算资源之上的统一逻辑模型，开发者可以在“标签”这种逻辑模型视图上结合画像分析、规则预警、文本挖掘、个性化推荐、关系网络等多个业务场景的数据服务模块，通过接口的方式进行快速的应用搭建。

这种方式的好处在于：

- 蔽掉应用开发人员对于下层多个计算存储资源的深入理解与复杂的系统对接工作；
- 通过数据服务的形式透出也有助于 IT 部门对数据使用的管理，避免资源的重复和冗余。

简单来说，因为大数据计算能力的增强，开发者只需要把需要使用的数据在模型当中进行管理后，即可通过 API 方式进行相应的计算对接到产品界面端上，或通过提供的界面配置功能直接生成可以独立部署的代码快速搭建相应的大数据产品。

整个产品系列包括以下几个模块：

- 标签中心
  - 云计算资源管理
  - 模型探索
  - 模型管理
- 整合分析
  - 分析服务
  - 界面配置

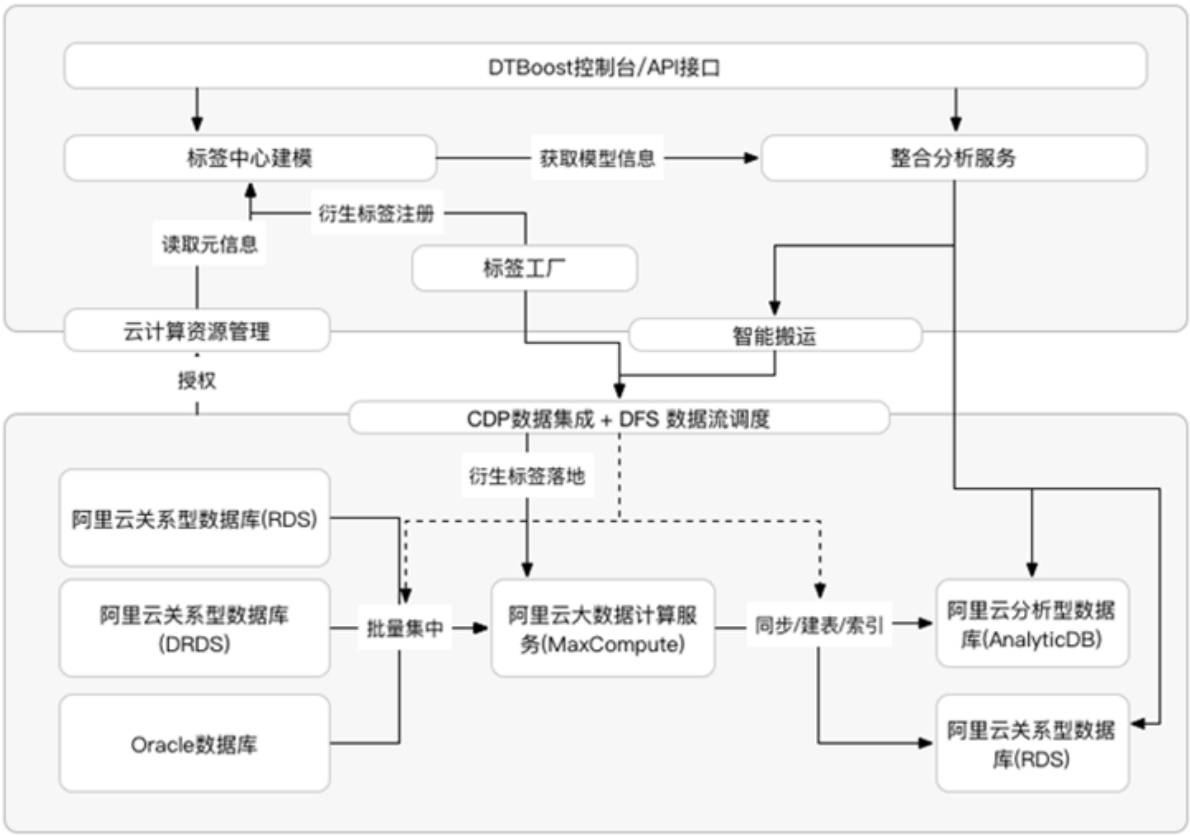
9.3 架构总览

在大数据环境下，一个数据应用往往需要通过多个计算资源来配合完成，最简单来说，一般数据需要先在离线环境当中进行离线加工处理（ETL），再同步至在线数据库当中进行在线分析查询（OLAP）。那么标签中心所能够做的就是与多个数据库进行通信，获取多个计算存储资源的数据元信息后进行逻辑建模，并把各个数据服务模块接口传入的指令解析后将真实的计算命令传给每一个计算资源。

下面以其中以 DTBoost 数据服务模块当中整合分析作为案例来解释总体的架构。

以最常见的 OLAP 分析场景来看，一般需要从业务库当中将数据进行抽取，加载到大数据（离线）计算服务 MaxCompute 当中进行集中，进行相应的加工、衍生后，再把所需要分析的数据同步到在线分析库（在大数据量下通常会使用分析型数据库 AnalyticDB）当中。

图 9-2: 标签中心技术架构



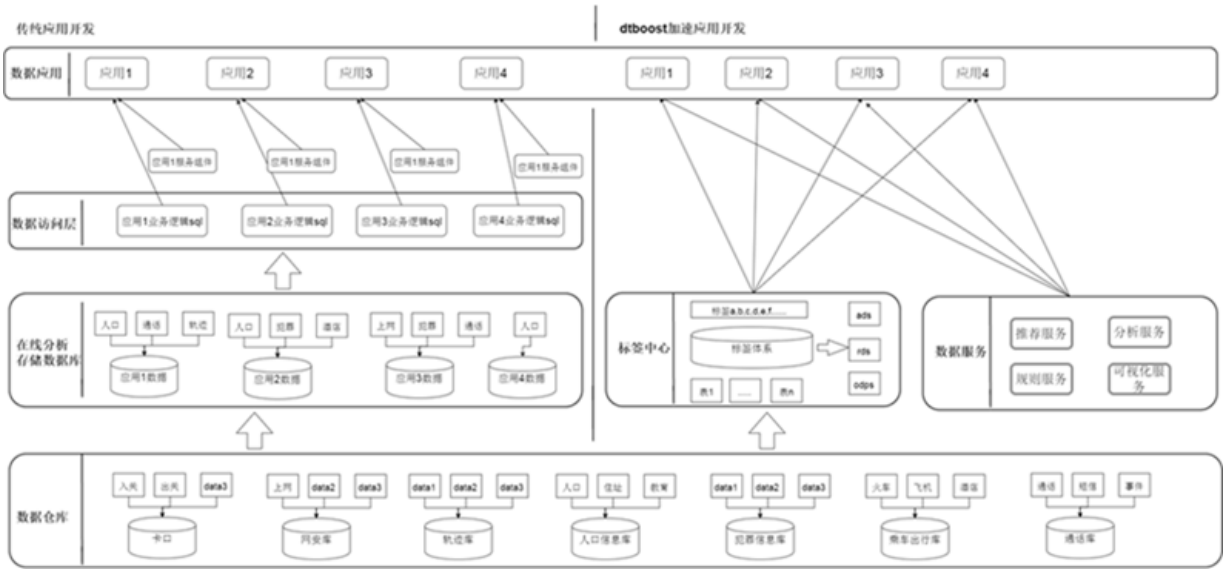
如图 *9-2:* 标签中心技术架构所示，用户从 DTBoost 控制台或 API 进入，通过把自己的云计算资源授权给 DTBoost 后，就可以通过 DTBoost 读取各个云计算资源中的数据元信息。经过建模配置后，在相关的数据服务模块中可以进行手工/自动触发标签中心的智能搬运模块，通过把相关的数据同步调度任务发送给数据流服务 DFS（Data Flow Service）和数据整合 CDP，来对所需要整合的数据以标签粒度来进行业务库到离线数据仓库的批量大集中，以及到在线分析数据库的同步、建表、索引工作。在数据准备完成之后，就可以通过相关的数据服务 API 接口或者在控制台上基于标签模型视图之上进行相关的计算。对于当中需要离线计算加工的部分，一些常用的加工可以通过标签工厂来对标签进行批量的衍生（如常见的聚合、筛选组合等）落地到大数据计算服务当中（MaxCompute）。

整个过程可以看做 DTBoost 在大数据平台之上对各个计算资源之间满足常见业务场景的架构方案进行了系统集成，简化了各个系统之间手工对接等过程。

9.4 场景概要

用户除通过控制台对各个模块进行配置操作以外，各个模块从数据元信息到数据服务的操作处理都可以透过开放API整合入自己的应用系统当中。这种服务化的方式一方面提高了系统整合的便利性，另一方面也对企业数据应用管理上提供了便利。

图 9-3: 加速开发流程



如图 *9-3:* 加速开发流程所示，从企业 IT 架构上来看，IT 或者数据部门可以通过 DTBoost 以数据服务化的方式把计算资源、数据资源、数据计算方法打包在一起，提供给业务部门开发、外部合作伙伴。一方面对应用开发者来说即开通即使用，方便快捷；另一方面从 IT 部门来说，对于平台的资源管控更加有效，一定程度上降低了数据的冗余存储与加工，特别针对于业务算法、消费者画像这些需要使用到明细数据计算的场景，既能够使用到明细数据，又不会影响到原始数据的生产，不造成大数据量的冗余拷贝，还能够降低数据使用的门槛，提供了有力的支撑。

9.5 功能模块

9.5.1 标签中心

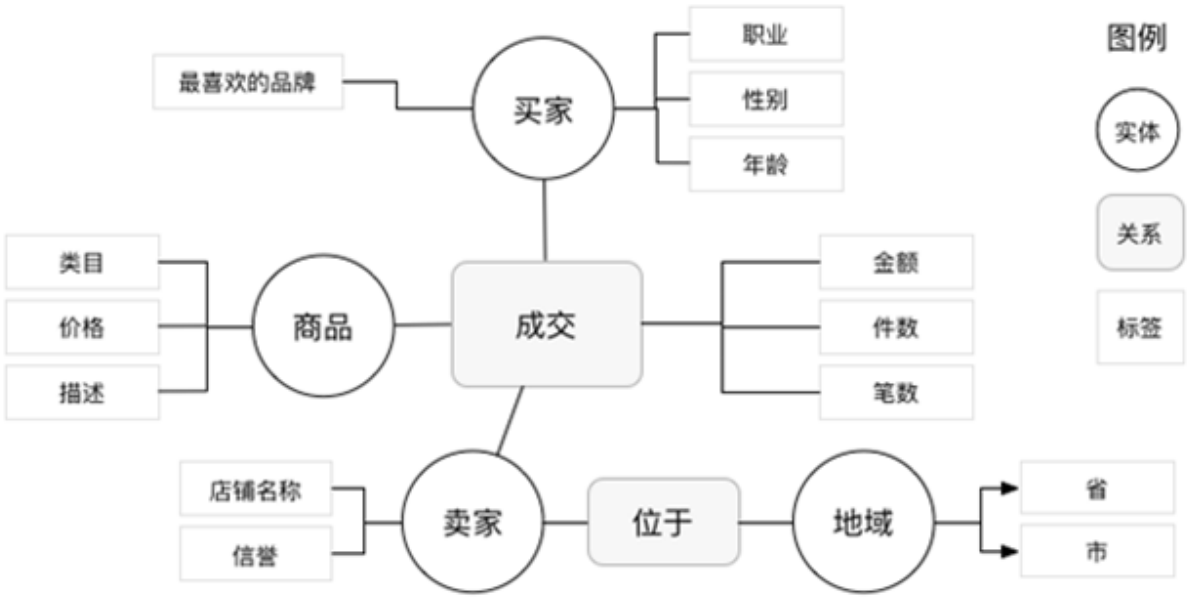
9.5.1.1 概念说明

标签中心的作用是在现有的数据表之上构建跨计算存储的逻辑模型，直接让用户在视图层上对数据进行管理、加工、查询，屏蔽下层的多个大数据计算存储资源，简化数据的使用。当整个数据架构越复杂，越是需要多个计算存储资源组合使用的场景下，标签中心的价值就越为明显。

标签建模的方法来源于阿里巴巴用户画像体系，广泛应用于精准营销、个性化推荐、用户画像、信用评分等需要基于明细数据进行计算的大数据应用当中。所谓标签就是对用户这一对象的一个最小描述单元，代表着所描述对象某一个具体的客观事实的抽象表达，如属性（性别，标签值男、女；年龄，标签值实际年龄），行为（成交金额、收藏次数、位置定位），或者是兴趣（对于多个关键词的偏好度），是一种以业务视角出发的数据建模方法，标签既可能是数值、也可能是枚举值，也可以是多个 Key-Value 组织的列，还可能是多字段组成的事实表（如对象、时间、谓语、宾语）。从概念模型上讲，标签体系就是围绕多个实体对象，如买家-卖家-商品-企业-设备，以及实体之间的关系，如成交-检修-位于等等，建立标签化描述的方法。

标签建模如图 *9-4:* 标签建模所示。

图 9-4: 标签建模



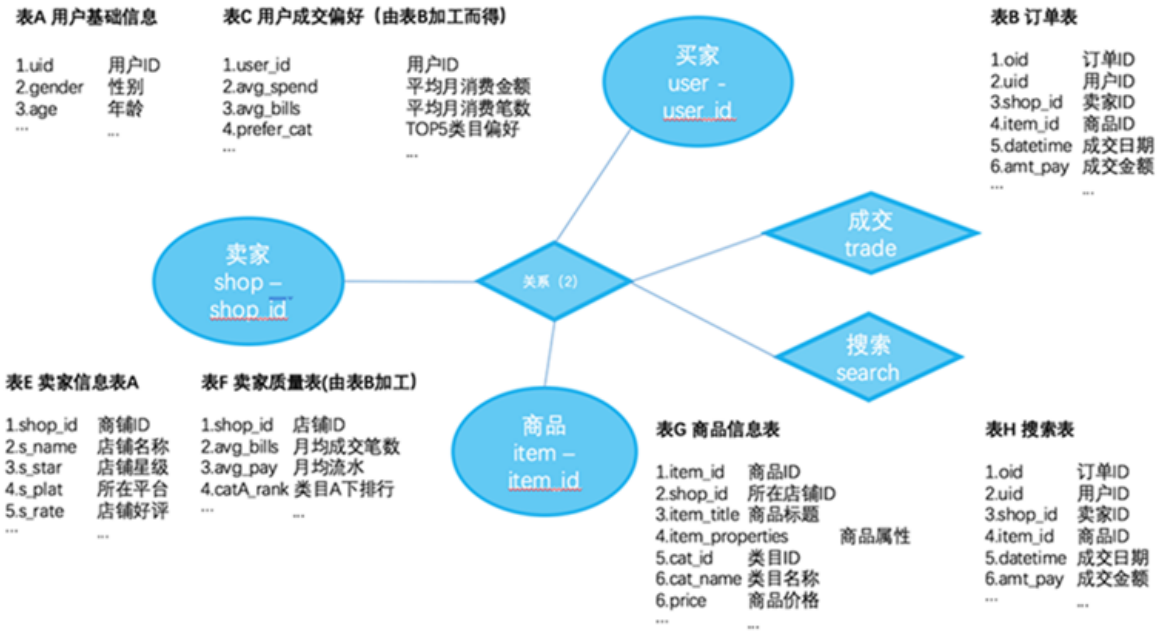
这种建模方式看起来可能类似于角模型（Anchor）或者是图模型（Graph），其实并不然。传统的建模过程是根据业务需求设计概念和逻辑模型，再根据逻辑模型对物理数据表进行加工和规整。而标签建模是在已有的物理数据/模型之上直接建立逻辑模型，通过各个数据服务的代理解析，让用户可以在视图上直接进行各类的计算，不需要预先对物理数据进行大规模的加工处理，即用即算。

但需要明确的是，总体来说，标签仍然是建立在物化数据之上，因为在跨计算的语境之下可能会面临多个计算的查询语言和性能的差异，建立在逻辑请求上的标签很可能会无法执行，所以总体来讲定义的每一个标签还是需要对应到落地的物理表上。但在 DTBoost 当中，可以在相应的数据服务当中以某一个计算查询逻辑定义为一个临时标签使用，但关系到跨计算之时还是需要将之物化，避免错误发生的可能。

标签模型v2.0是围绕实体（Entity）、关系（Link）、标签（Tag）三大元素对分布在不同数据库中的数据进行网络化的建模方式。实体用于描述某个客观的对象，如设备-人员-地址等，对应到物理数据表上一般就是属性表，有一个主键来代表每一个对象，剩下的每一列就是标签即描述对象的属性。那么关系是表示对象和对象之间的联系、事件、行为，一般对应到物理数据表上一般就是事实流水表，如成交-检修-乘车等。

实体关系建模如图 *9-5:* 实体关系建模所示。

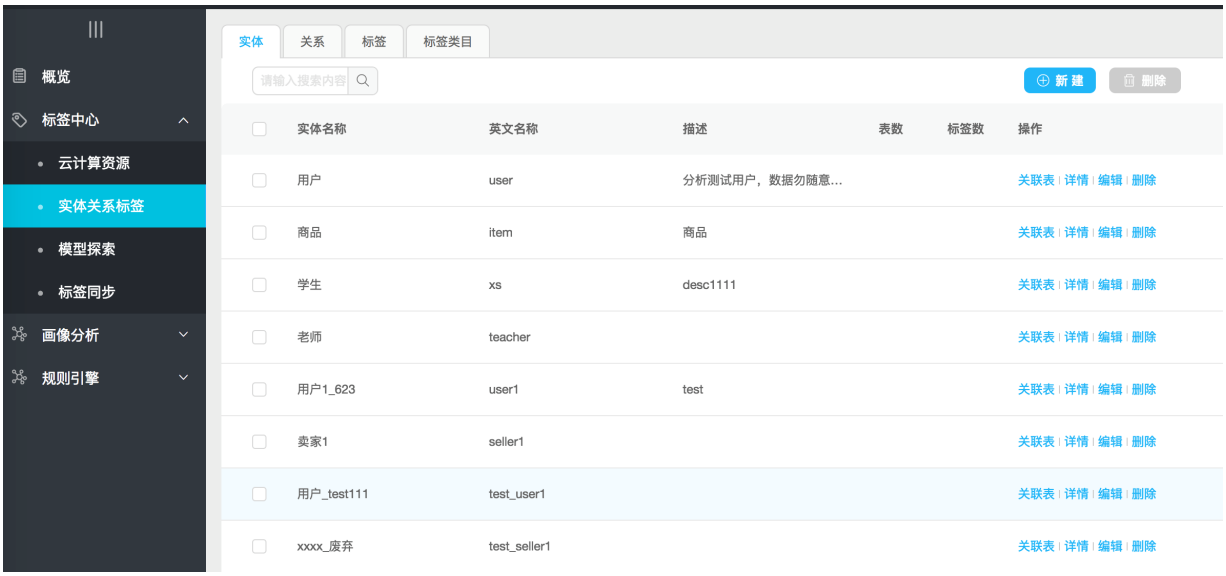
图 9-5: 实体关系建模



相比于指标-维度体系，这种建模方式更适用于对于明细数据描述和表达。明细数据很大一部分都是事实表，引入关系的概念对应到流水事实表上，把多个实体之间的关系很好的呈现表达，既有利于管理也方便分析时的表达，在对业务端呈现上也更接近于概念模型的设计一样可被一般人理解。

在经过建模转化之后，可以将上表中的模型逻辑关系转化为图 *9-6:* 实体关系管理所示。成交表对应到关系结点上，金额和时间是关系上的标签，用户表和商品表对应到买家和商品两个实体上，性别、年龄是买家的标签。这种建模方式非常便于各类基于明细行为、关系数据进行分析的场景。

图 9-6: 实体关系管理



您可以在标签中心页面下看到标签中心的几大功能，包括模型管理、云计算资源管理和模型探索。

9.5.1.2 适用场景

标签中心是跨计算存储、可在物理模型之上逻辑动态建模、与数据服务结合面向大数据应用开发的数据建模、数据管理工具，并能够通过可视化的方法清晰的展现企业的数据模型视图。

标签中心适用于以下场景：

- 数据模型探索管理

  标签中心提供一种业务视角的数据发现、模型探索的工具，便于业务人员、开发人员、数据管理人员透视企业的数据资产。

- 为数据服务提供视图支撑

  为多个计算引擎上的数据提供一个统一的数据视图，结合数据服务能够方便的进行业务逻辑计算操作

- 数据权限管理

  可以通过逻辑层对数据访问权限进行有效控制，比物理表的访问管理更加安全有效

9.5.1.3 功能组件

9.5.1.3.1 云计算资源管理

云计算资源管理就是支撑与多个计算存储资源通信，与元信息获取的基本功能模块。

目前 DTBoost 支持与以下计算存储资源的管理：

- Oracle 数据库
- 阿里云大数据计算（MaxCompute）
- 阿里云分析型数据库（AnalyticDB）

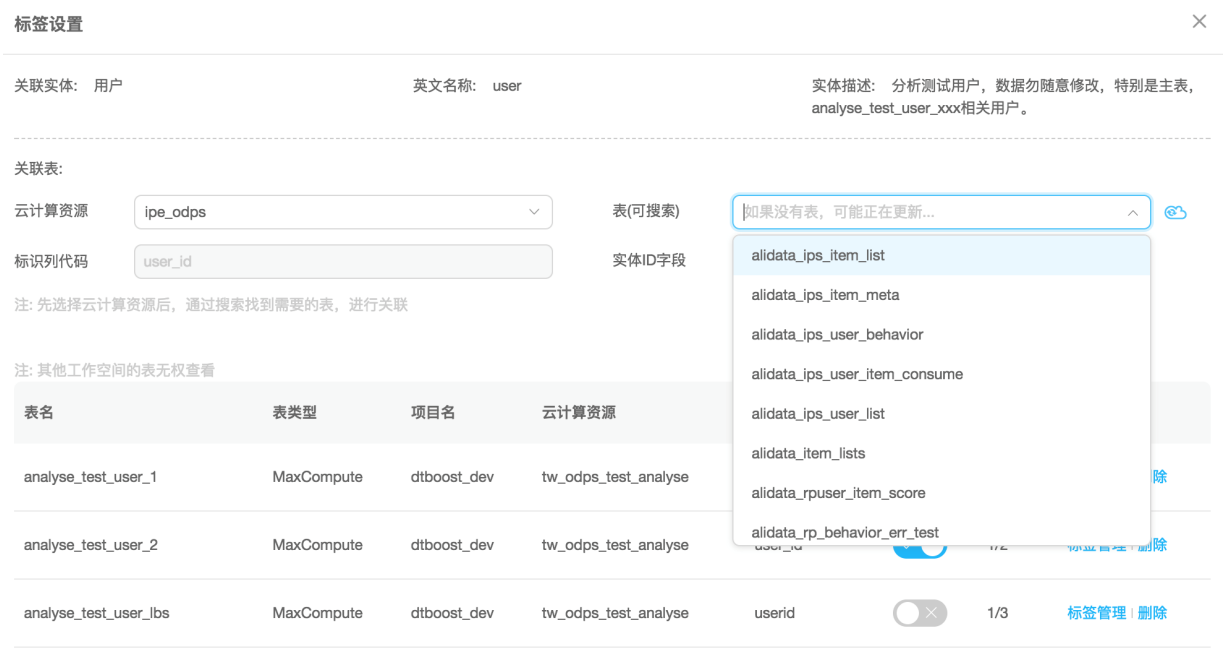
9.5.1.3.2 模型管理

实体关系管理

实体/关系管理是标签中心当中对逻辑模型进行配置的主要功能，能够读取不同来源的数据库的元信息，整合为实体或者关系。

描述同一个实体（主键）的多张表可以在逻辑层上聚合在一个实体下，形成一张**大宽表**，如图 *9-7: 实体关系建模示意*所示。

图 9-7: 实体关系建模示意



关系的建立则是可以把联合主键表看作为关系，将多个实体关联起来。其余的描述字段则根据相应的情况定义为标签，如图 *9-8: 关系定义*所示。

图 9-8: 关系定义

标签管理

标签管理模块能够对所有的标签进行查看、检索和修改，如图 *9-9:* 查看实体详情和图 *9-10:* 设置标签所示。

图 9-9: 查看实体详情

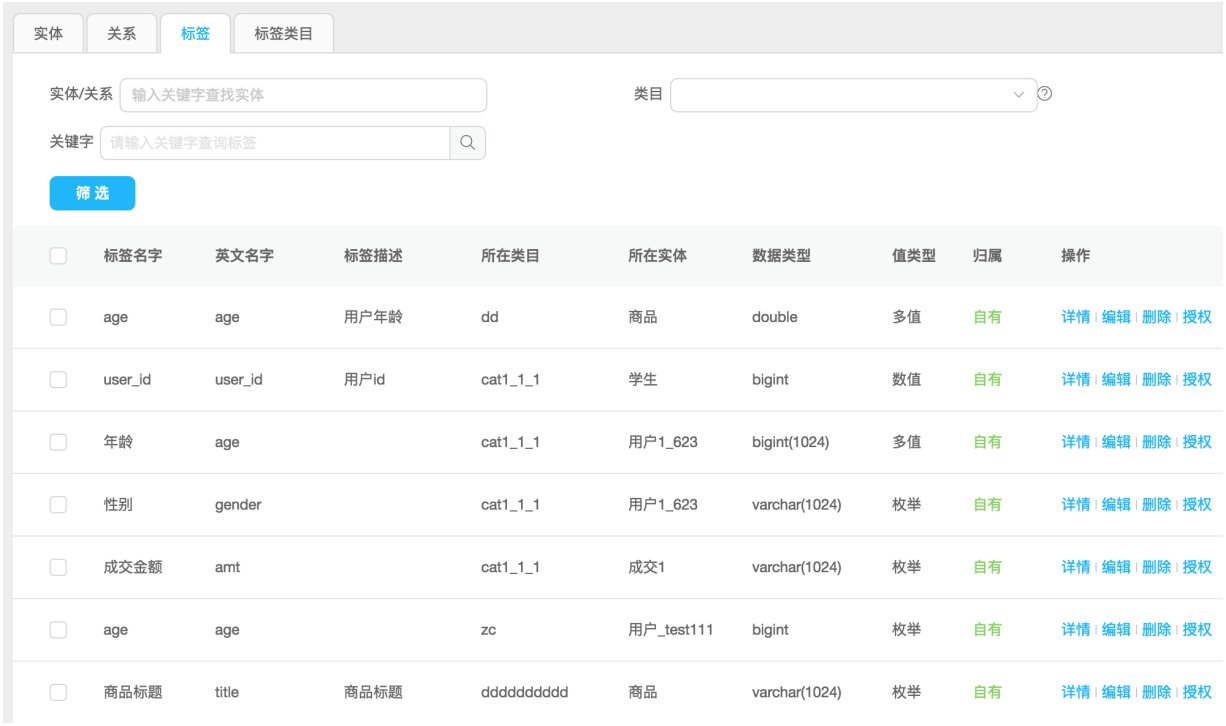
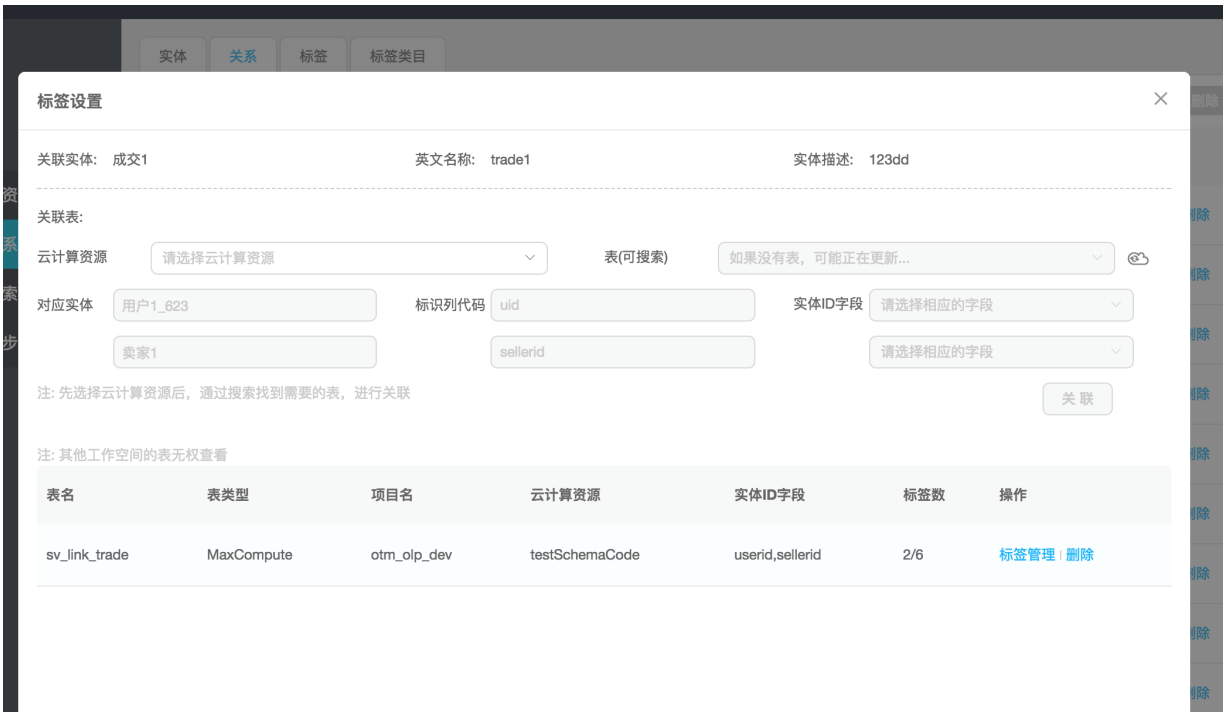


图 9-10: 设置标签

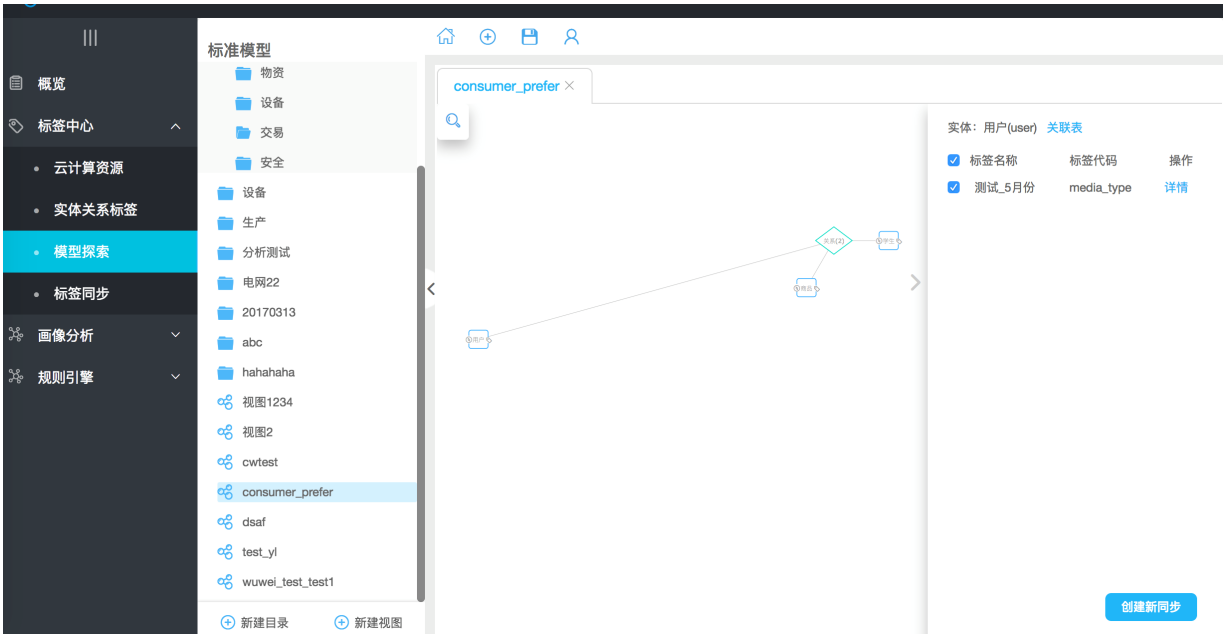


9.5.1.3.3 模型探索与数字订阅

模型探索部分可以通过关系图的方式查看所有的实体，实体与实体之间的联通关系及其属性，以及实体/关系下关联的标签情况。

如图 *9-11: 实体关系模型探索*所示，通过模型探索可以对整个标签模型进行全局的分析查看。

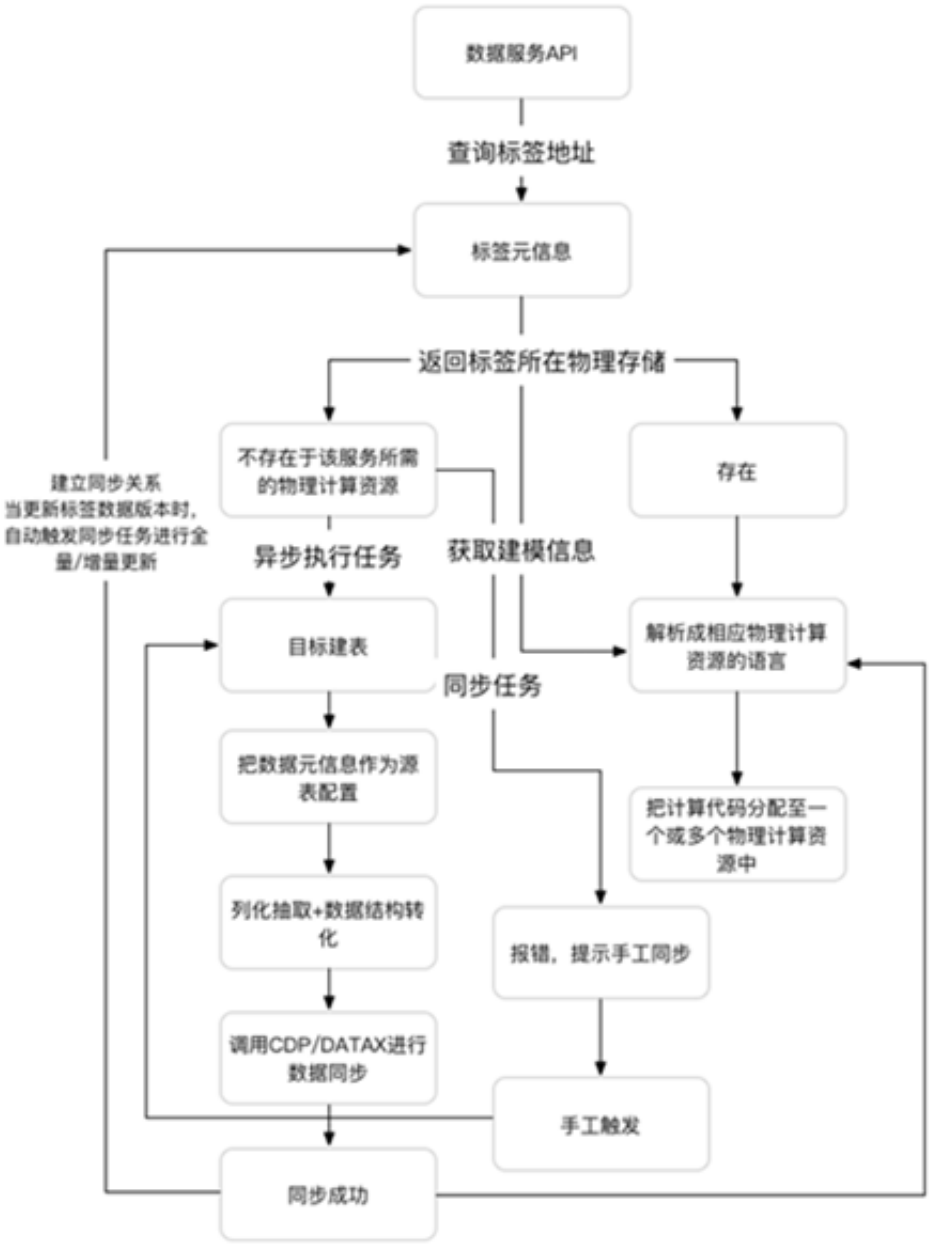
图 9-11: 实体关系模型探索



标签数据订阅是 DTBoost 处理跨计算数据流转的重要功能之一。在相应的数据服务需要使用到数据的时候，标签中心提供了将分散在多个存储当中的数据订阅至数据服务需要计算的位置的功能。对于同步且相应时间要求高的场景来说，需要用户在相应的数据服务当中进行提前的手工订阅操作，对于异步或者请求相应要求不高的同步的计算场景来说，这个订阅过程对于用户来说透明。

标签中心的技术架构如图 *9-12: 标签中心技术架构*所示。

图 9-12: 标签中心技术架构



智能搬运内置了针对几套典型的架构路径：

- 批量大数据在线分析，如图 *9-13:* 批量大数据在线分析所示

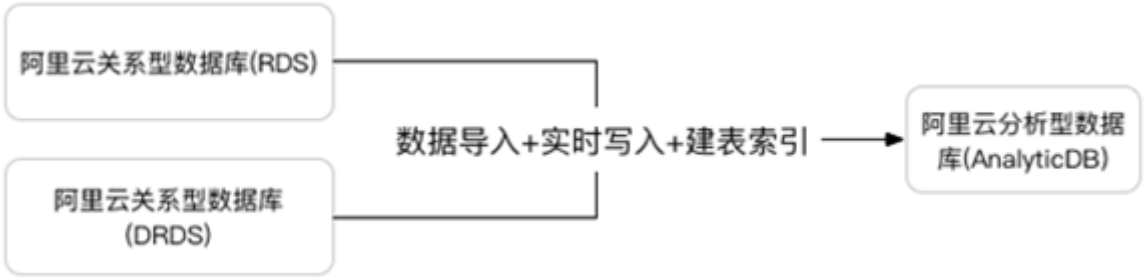
图 9-13: 批量大数据在线分析

- 批量数据算法计算在线查询，如图 *9-14:* 批量数据算法计算在线查询所示

图 9-14: 批量数据算法计算在线查询

- 实时大数据在线分析，如图 *9-15:* 实时大数据在线分析所示

图 9-15: 实时大数据在线分析



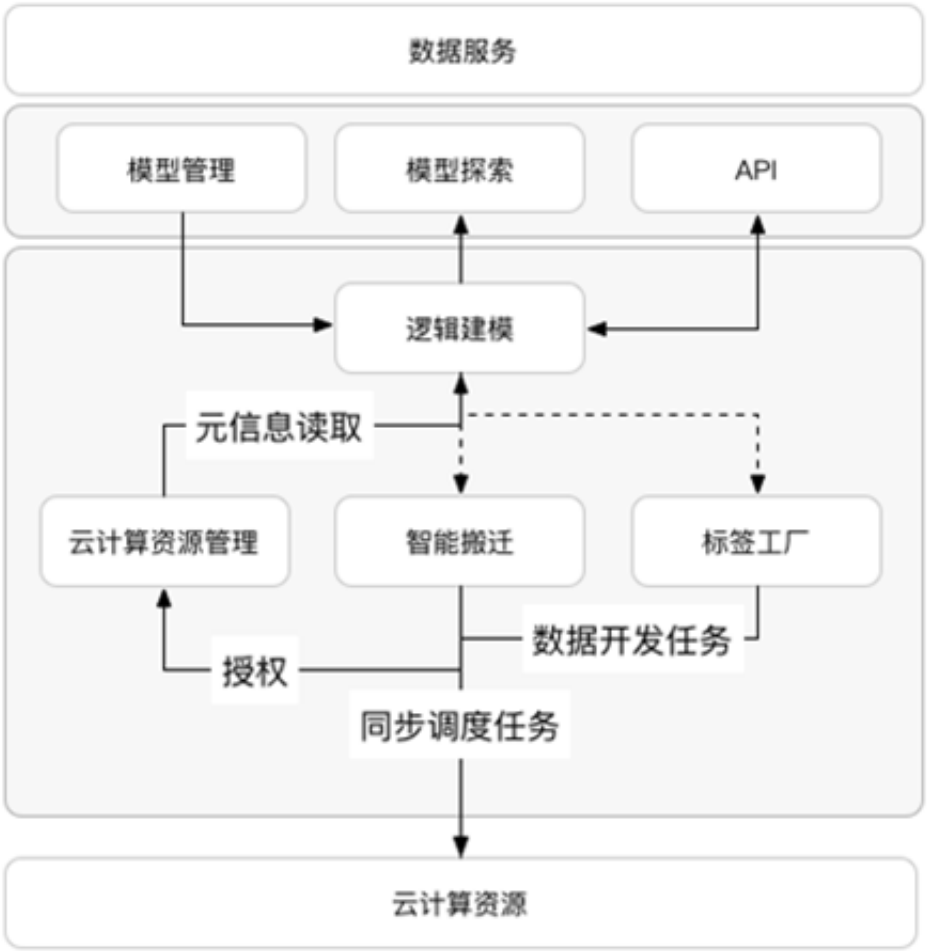
  - 对于整合分析这类 OLAP/ADHOC 场景来说，提供了将 Oracle，关系型数据库（MySQL）等业务库中的数据同步至大数据计算（MaxCompute）中，再订阅到所使用的分析库中，如阿里云分析型数据库（AnalyticDB）等。

- 对于规则引擎这类流式计算的场景来说，提供了将离线数据、流式数据进行归并，将规则所需要的离线历史数据订阅至阿里云表格存储当中，并根据规则计算结果订阅至所需要的存储计算资源（MySQL/MaxCompute/AnalyticDB等）中。
- 对于目前尚未以标准方式提供的订阅路径，可以进行相应的定制。

9.5.1.4 技术架构

技术架构如图 *9-16:* 标签中心组件架构所示。

图 9-16: 标签中心组件架构



9.5.1.5 产品特性

对于这种数据体系的规划上来说，往往是由业务驱动的，是累积增加的，随着不同的业务板块的开展会逐渐纳入更多的数据源。如果按照传统数据仓库的做法会面临几个问题：

1. 需要不断地在物理层进行数据表的归并，下层表的频繁变化可能会造成数据使用的不稳定；

2. 当标签的需求越来越多，因为不可能无限制的在物理层将数据拼在一张宽表当中，那分散的数据表也会越来越多，会造成检索和管理的困难。
3. 在不同的应用当中不可能是整表使用，往往是需要多张表中的某几列，那多个应用不断的抽取再整合也会造成管理和检索的困难。
4. 标签可能是实时数据、也有可能是离线数据，数据存储方式不同同样造成管理使用的困难。

由此，标签体系建模和传统BI分析建模有几大特性：

- 业务视角管理

  围绕实体-关系-标签这三个元素进行建模，是从业务的角度出发对数据进行组织管理，而不是从表的概念出发进行建模，便于应用层对数据运用和管理的理解、操作，以近似于概念模型的形态透出，让人人都能看得懂。

- 跨计算的统一逻辑模型

  传统建模的数据来源和模型的使用一般在同一数据库当中，而大数据环境下因为数据采集类型的多样性，和数据计算的多样性使得来源和使用分散在不同的计算存储资源当中，数据产生与加工首先就可能分布在不同的数据库当中，其次同一份数据需要进行跨流式、Adhoc 类多维分析、离线算法加工等多种方式的计算，数据需要能在多个存储和计算资源当中自由流转。

  所以标签体系是把多个计算当中的拷贝在逻辑视图上进行唯一映射，即一个标签对应到多个计算当中的物理字段。

- 灵活拓展性

  呈上，表/标签之间的逻辑关系的建立也是在逻辑层上完成的，这就使得模型的维护是可以动态建设的，便于模型的维护和管理，而无需在物理层将数据进行归并后再使用。每一个标签之间可以独立使用，这种离散的列化操作方式也使的数据的使用上更为灵活。

从另一方面来说，计算能力的增强和数据使用场景的丰富，更多的数据计算是需要直接作用在明细的行为数据上，而非只是对指标的多维统计。传统数据集市建模的“指标-维度”体系就略显狭窄。标签的定义上涵盖了多种数值类型，既可以是单列，也可以是维度 + 标签组成的复合标签（这种方式通常用于描述某种行为），赋予应用操作上更大的灵活度。

9.5.2 整合分析

9.5.2.1 适用场景

数据服务是架设在标签视图之上的业务功能模块，可以通过界面化的配置或者 API 的操作能够以标签为里度对跨计算资源的数据进行统一的业务计算操作。透过数据服务+逻辑建模的组合，既节

省工作量又有很好的扩展性。特别是对于大数据环境需要整合多个系统数据的前提下，很难一次把所有数据需求全部规划完整，那么这种动态逻辑建模的方式就有非常好的扩展性。从应用的角度来说，由标签模型视图层隔开明细数据复杂的数据结构，能够在相对扁平的标签体系之上进行明细数据上计算和查询，对于数据开发与应用开发的分工流程上来说也更为合理。

图 9-17: 整合分析

如图 *9-17:* 整合分析所示，整合分析作为 DTBoost 数据服务的其中之一，其所适用的场景主要是结合阿里云分析型数据库（AnalyticDB），将您分布在多个存储资源的数据整合起来，在标签模型上构建大数据画像类的交互式分析应用，让您的业务人员可以自由灵活的分析这些对象各种属性与行为之间的关联性。可以广泛应用于工业设备画像分析、企业经营画像分析、用户行为画像分析等多个场景当中。

大数据画像类分析应用有如下几个特性：

- 基于行为等明细数据的分析

  在过去以各项 KPI 指标计算为主要分析目的背景下，很容易把所有的指标计算提前构建。随着数据采集和使用场景的丰富，业务人员希望能够自由地分析各类行为明细数据，如查看不同客户属性在各个商品类目下消费的偏好和关联购买的情况，或者不同时间采购的不同类型、属性、地域设备的故障率与检修情况，还能够把多个维度细分下的具体客户/设备清单进行查看。业务人员进行的分析可能是任意维度之间的交叉关系，就很难进行预先的计算。

- 从半结构化数据中抽取特征

从另一点来说，灵活分析还意味着能够与预测、评分、文本特征提取等算法技术相结合，进行广度与深度兼备的分析。往往很多的画像特征如抽象的兴趣，如喜欢动漫、爱美一族等风格兴趣偏好类的特征，通常需要通过算法从用户的点击、收藏、购买行为与相关物品的文本描述当中进行特征抽取。这就需要能够借助一些偏好计算、文本挖掘类的算法能够从这些半结构化的数据当中对用户互相的特征进行深度的挖掘。

- 交互式的查询分析

  业务人员希望得到的分析是在数据当中探索有用的信息，如发现影响消费者购买的可能因素，或者故障设备的关联因素，这就需要能够根据不断调整的筛选条件、维度组合、下钻上聚能够快速返回结果，直到获取到足够多的信息。这就对查询速度的高响应提出了要求。

  在这种交互式的分析场景下也对整体界面的组织提出了要求，业务人员关心的是在不断探索中获得的数据洞察，如果还需要用户进行复杂的报表配置或者是数据结构/技术上的学习理解，就会大大影响数据探索发现的过程。各种数据的分析还需要与各种类型的可视化形态结合，除了常规的图表外，可能还需要各种尺度特别是城市内尺度的地图图表，表达拓扑关系的关系网络图表，以及能展示文本特征的图表。

从以上几点来看，交互式数据分析产品的开发变得非常有挑战性，应用开发当中既需要充分理解数据结构，才能把跨表查询的逻辑与界面交互进行有机的组织；还需要了解多个专业存储与计算资源的特性，把不同计算产出的结果组织到同一个分析界面当中；也需要熟悉各类的可视化图表与分析控件的使用方法，结合到不同类型的分析当中。DTBoost 整合分析模块就是面向这类场景为您提供以下功能，帮助您加速应用开发的难度。

9.5.2.2 功能组件

整合分析提供了两大块的功能，分析服务层部分用户可以在控制台中完成。

9.5.2.2.1 接口调试

分析服务接口模块可以让您在此进行分析语句的调试和自助化封装数据分析接口。画像分析的查询表达都是建构在实体关系模型之上的。

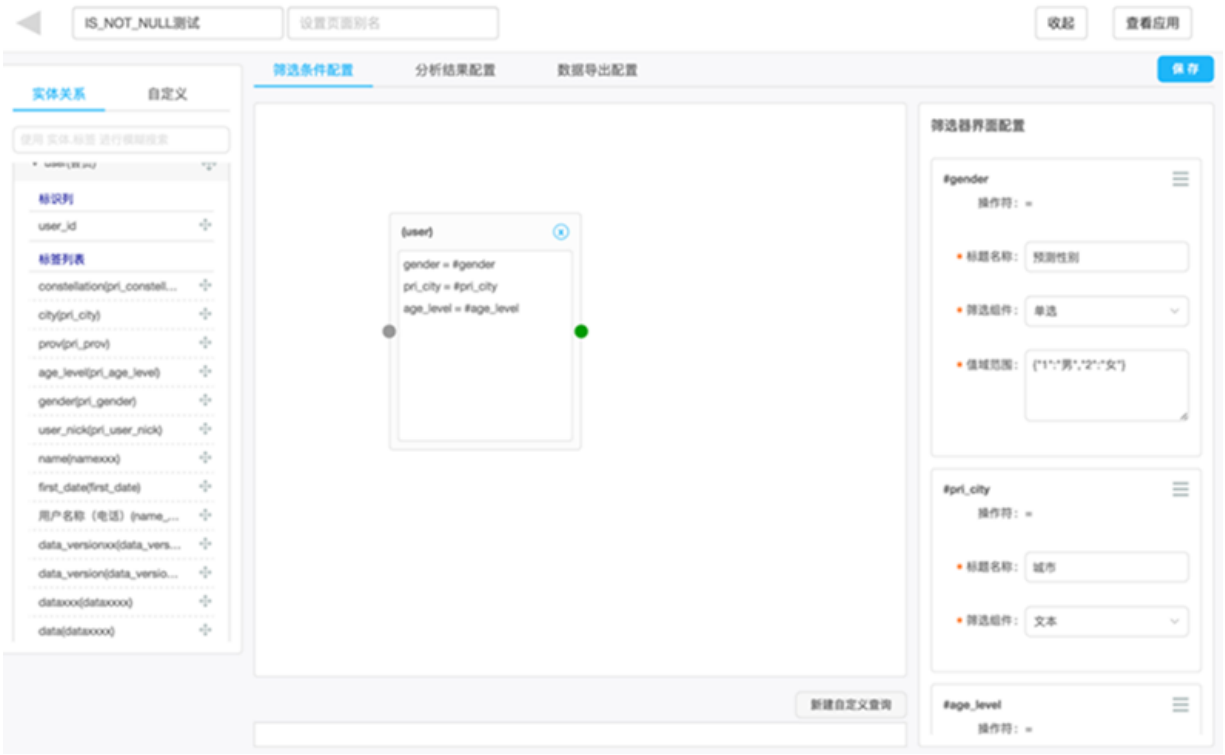
您也可以对接口进行调试，查看查询的结果/执行错误、语法每一步解析所耗费的时间以及所解析的真实 SQL 语句，来帮助您调试分析接口。

9.5.2.2.2 界面配置

如图 *9-18:* 筛选条件配置所示，通过画像分析界面搭建工具，灵活配置交互式画像分析界面。对筛选出来的特定的分析对象进行多维透视，并进一步钻取分析，并可以将分析筛选出来的对象导出到

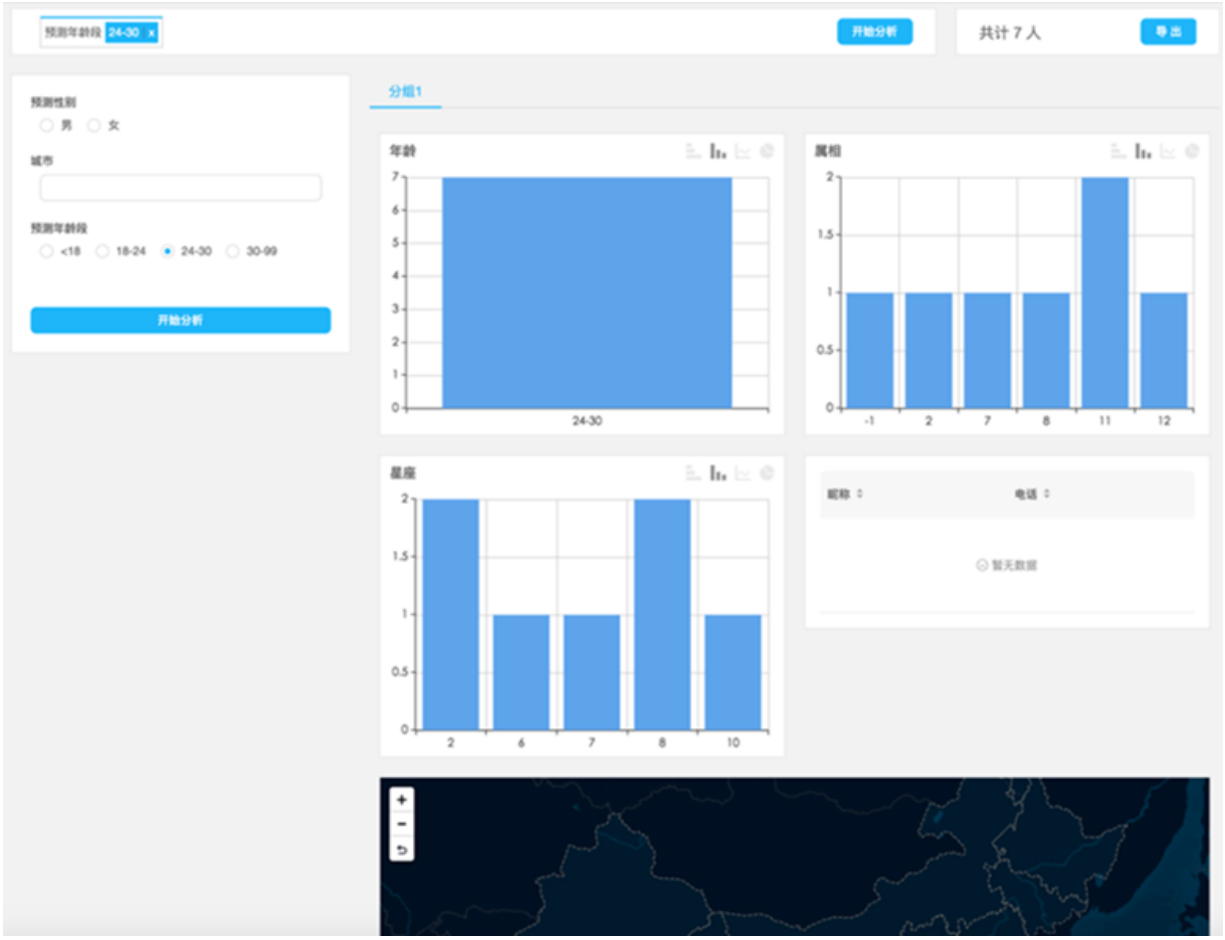
其他系统当中，如结合广告投放系统进行精准营销。整个界面的代码是完全开放的，可以无缝与您的现有系统进行整合。

图 9-18: 筛选条件配置



如图 *9-19: 分析应用*所示，上层提供的交互分析应用框架是以源代码的方式提供，用户只需把整个应用运行，会根据配置文件的填写自动渲染出一个可进行交互式分析的界面。用户可以进行代码的修改进行样式的改造。多个配置文件可以通过配置从不同的 URL 进行路由，让不同的用户可以看到不同的分析应用。

图 9-19: 分析应用



9.5.2.3 典型应用

9.5.2.3.1 用户全景画像

在受众分析、CRM、用户行为、人口分析等场景下，通常需要对这些人群的明细行为数据进行分析。

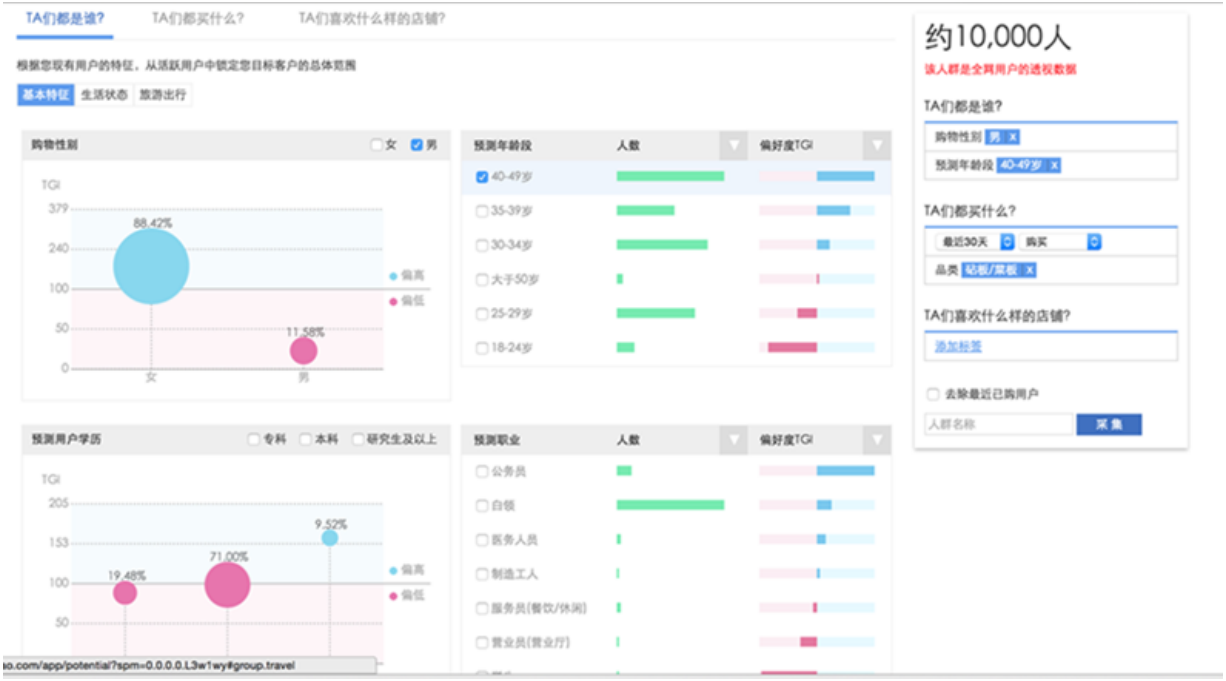
分析人员在进行分析的时候，可能多种行为与多种用户属性之间的组合筛选变化多样，无法按传统数据分析建模方法提前预算好所有组合。

例如，用户根据消费者行为选取想要分析的目标群体，如图 *9-20:* 选择目标群体所示。

图 9-20: 选择目标群体

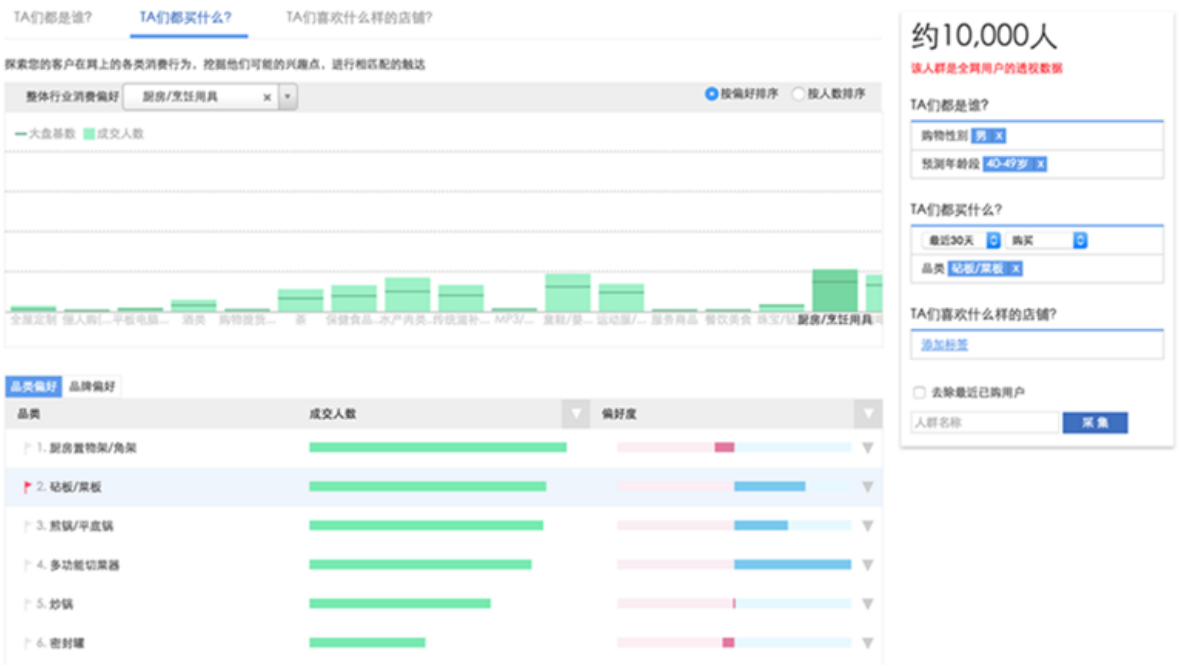
对于选取的目标群体，分析其各项特征，发现分布密集的特征，如图 *9-21:* 分析各项特征所示。

图 9-21: 分析各项特征



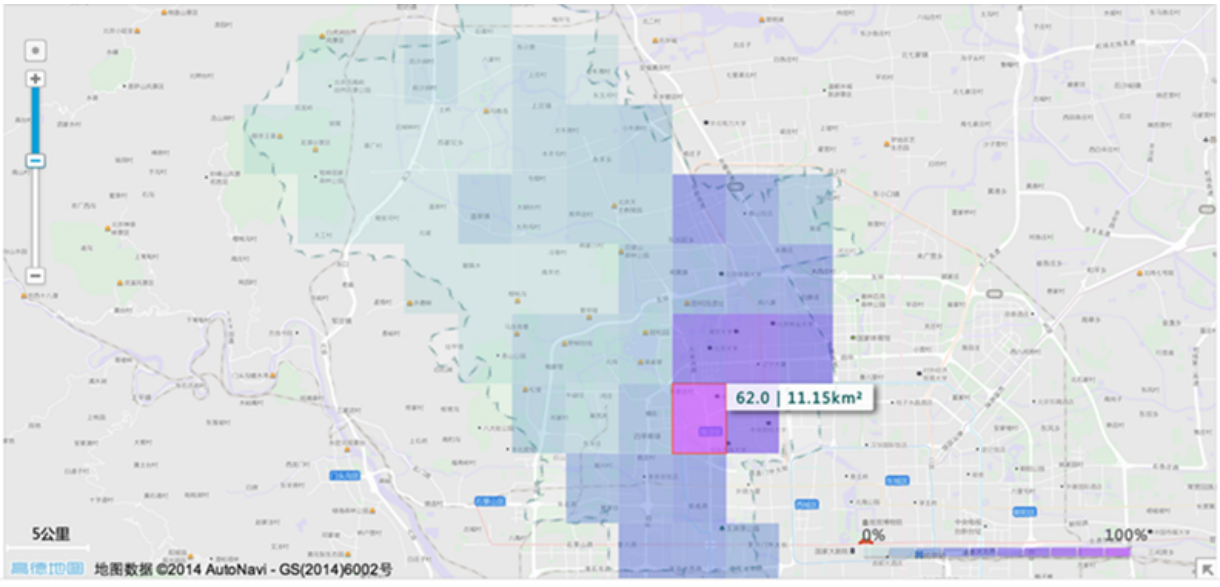
再根据这些典型特征，找到还未购买的群体，如图 *9-22:* 找到未购买的群体所示。

图 9-22: 找到未购买的群体



针对这类群体，对接下游系统，如图 *9-23: 对接下游系统*所示。

图 9-23: 对接下游系统



9.5.2.3.2 设备全履历

例如，设备全履历如图 *9-24: 电压等级*、图 *9-25: 地级供电公司*和图 *9-26: 设备明细*所示。

图 9-24: 电压等级

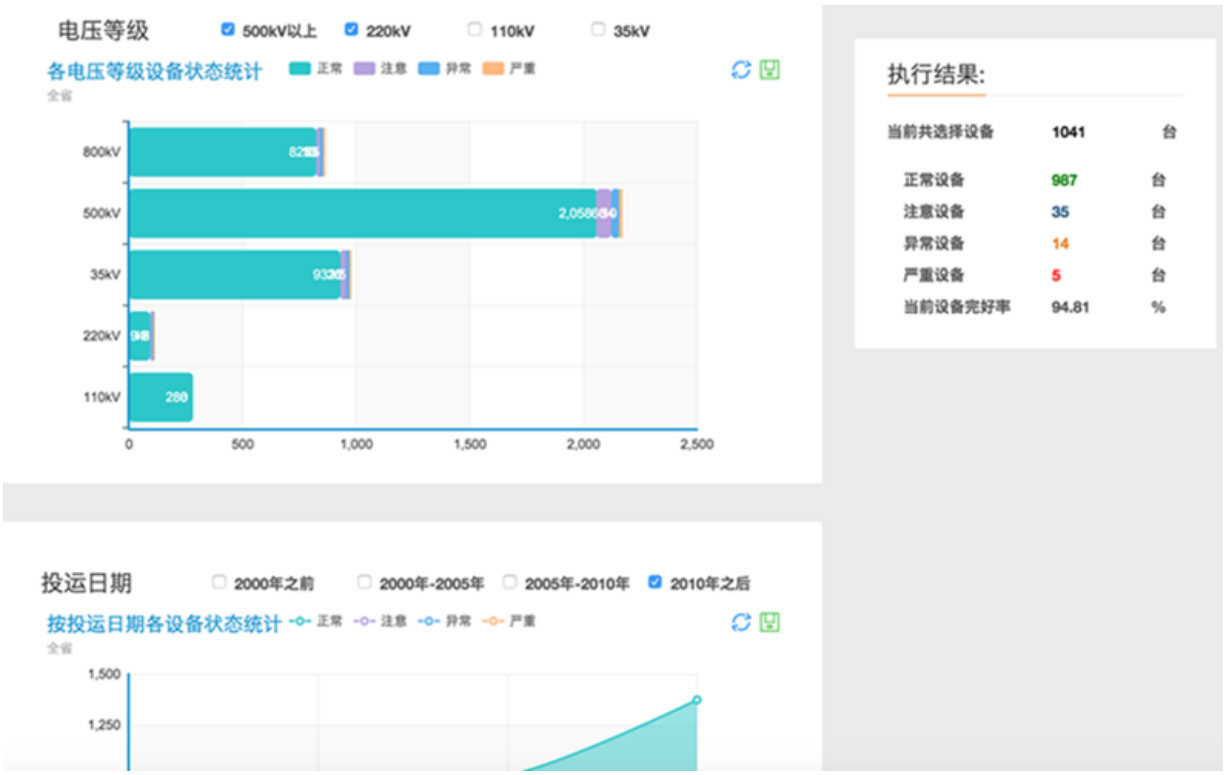


图 9-25: 地级供电公司

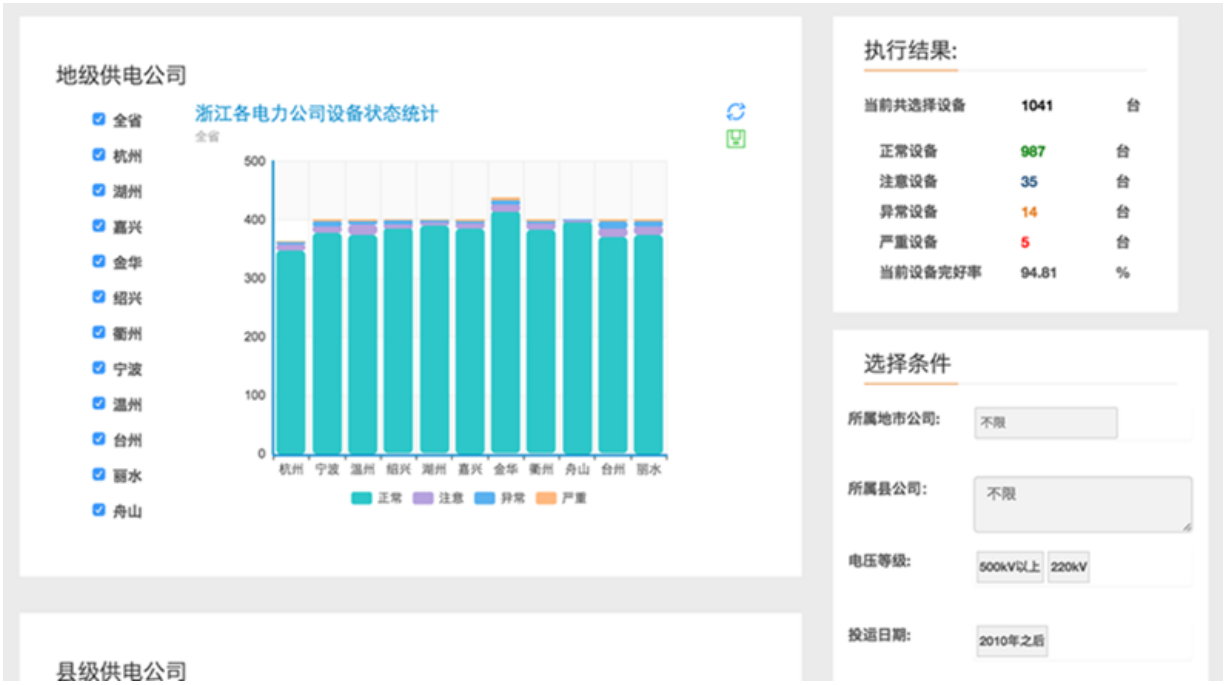


图 9-26: 设备明细

序号	设备编号	电压等级	生产厂家	设备名称	投运年份	所属公司	设备型号	所属区县	设备类型	设备状态	消缺日期
1	GLKG10605	220kV	厂商E	隔离开关10605	2000/03/19	杭州	10605	临安市供电公司	隔离开关	严重	-
2	GLKG10641	220kV	厂商A	隔离开关10641	2001/02/17	杭州	10641	建德市供电公司	隔离开关	严重	-
3	GLKG11603	110kV	厂商C	隔离开关11603	2016/04/14	湖州	11603	德清县供电公司	隔离开关	严重	-
4	GLKG11604	110kV	厂商D	隔离开关11604	2010/06/15	湖州	11604	湖州市本市局	隔离开关	严重	-
5	GLKG12601	35kV	厂商A	隔离开关12601	2007/03/18	嘉兴	12601	桐乡市供电公司	隔离开关	严重	-
6	GLKG13602	35kV	厂商B	隔离开关13602	2004/02/17	金华	13602	东阳市供电公司	隔离开关	严重	-
7	GLKG14609	35kV	厂商D	隔离开关14609	2008/06/15	丽水	14609	庆元县供电公司	隔离开关	严重	-
8	GLKG14610	35kV	厂商E	隔离开关14610	1996/06/18	丽水	14610	龙泉市供电公司	隔离开关	严重	-
9	GLKG14627	35kV	厂商B	隔离开关14627	2015/06/14	丽水	14627	景宁县供电公司	隔离开关	严重	-
10	GLKG14628	800kV	厂商C	隔离开关14628	2009/06/15	丽水	14628	松阳县供电公司	隔离开关	严重	-

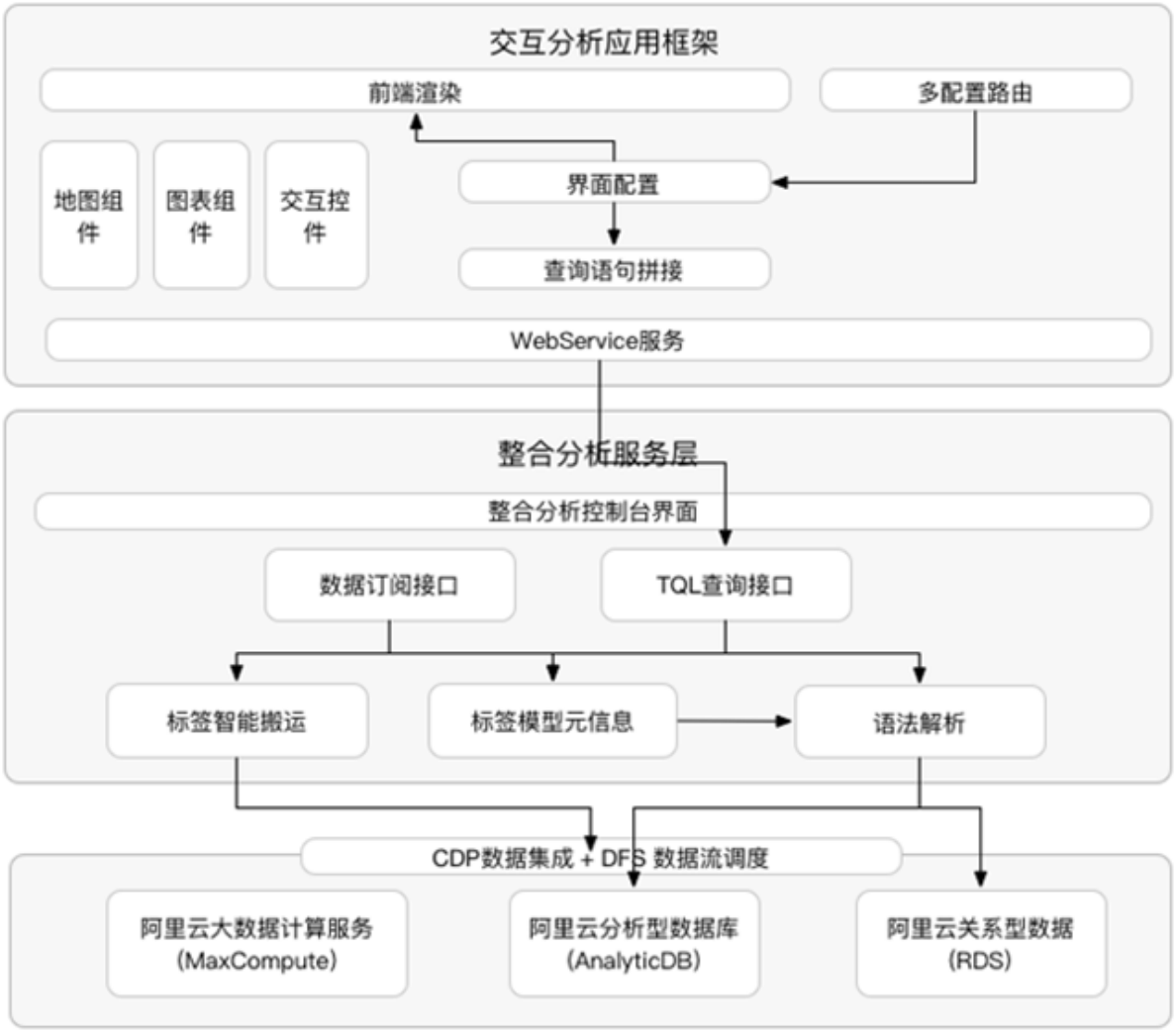
总数：1 / 3 页

首页 | 上一页 | 下一页 | 尾页

9.5.2.4 技术架构

整合分析的技术架构如图 *9-27:* 整合分析技术架构所示。

图 9-27: 整合分析技术架构



从技术架构来讲分为两个部分，在服务层部分通过 TQL 查询接口接收用户输入的参数，结合标签模型中的元信息会进行真实 SQL 的语法解析，然后把相应的 SQL 传送给相应的计算资源进行计算，通过接口获得返回的计算结果。使用 Debug 模式同时返回所解析的真实 SQL 以及每一步计算所耗费的时长，可以用于优化相应的查询语句。此处的语法解析主要是把用户在扁平化的标签模型视图上的查询逻辑，翻译为真实的多个表之间关联 JOIN 的查询。

数据订阅部分可以通过界面实现数据的一键搬迁，控制台界面会调用数据订阅的接口获取相应数据的元信息，调用标签中心底层智能搬运的接口，在相应的计算存储资源会自动建表建立索引后，触发调度任务进行数据同步。

在交互分析的应用框架层，会提供应用开发配置框架的源代码。其中包括相应的前端组件、WebService 后端服务、界面配置文件以及根据界面配置文件渲染相应界面同时翻译为相应的查

询的接口。同时配置文件还能够进行相应的路由配置，让不同的页面 URL 可以路由到不同的配置，让不同的人看到不同的界面。

9.5.2.5 产品特性

DTBoost 具备如下特性：

- 一键数据整合

  针对不同的分析主体，您可一键完成分布在多个存储资源当中的多个标签到在线查询数据库当中的同步、索引工作，像管理一张表一样的管理不同数据源。兼容的在线分析数据库既可以是阿里云分析型数据库（AnalyticDB），也可以是阿里云 RDS 关系型数据库。

- Web 开发友好

  Web 应用开发者者直接通过与整合分析查询和标签元信息 API 接口的交互，结合阿里云 DataV 或是其他图表组件，即可以快速搭建自己的分析型数据产品。

- 查询表达简单

  在扁平化的标签体系上，一定程度简化了表关联和子查询的表达，让 Web 应用开发人员更加关注在应用逻辑而非数据表的组织逻辑。查询参数提供 JSON 对象模式，也提供与 SQL 相似的的 TQL（Tag Query Language）模式。

- 与 DTBoost 其他模块无缝结合

  由于标签体系下，多个模块之间共享同一个标签视图，以及同一个标签在不同的存储计算资源能够自动搬迁，使得整合分析能够与 DTBoost 的算法模块、特征工程模块、实时预警模块产出的数据有一致性的表达，相互打通无缝结合。

- 交互式分析应用框架

  以 SDK 代码的方式提供分析界面配置工具，即刻生成交互式的分析应用。相比传统BI工具，配置出来的分析界面像一个独立的交互分析产品，可以整合入您整体的分析系统当中，更容易让用户灵活的洞察，对实体的属性、行为、地理出行等进行灵活的分析。

10 大数据管家

10.1 什么是大数据管家

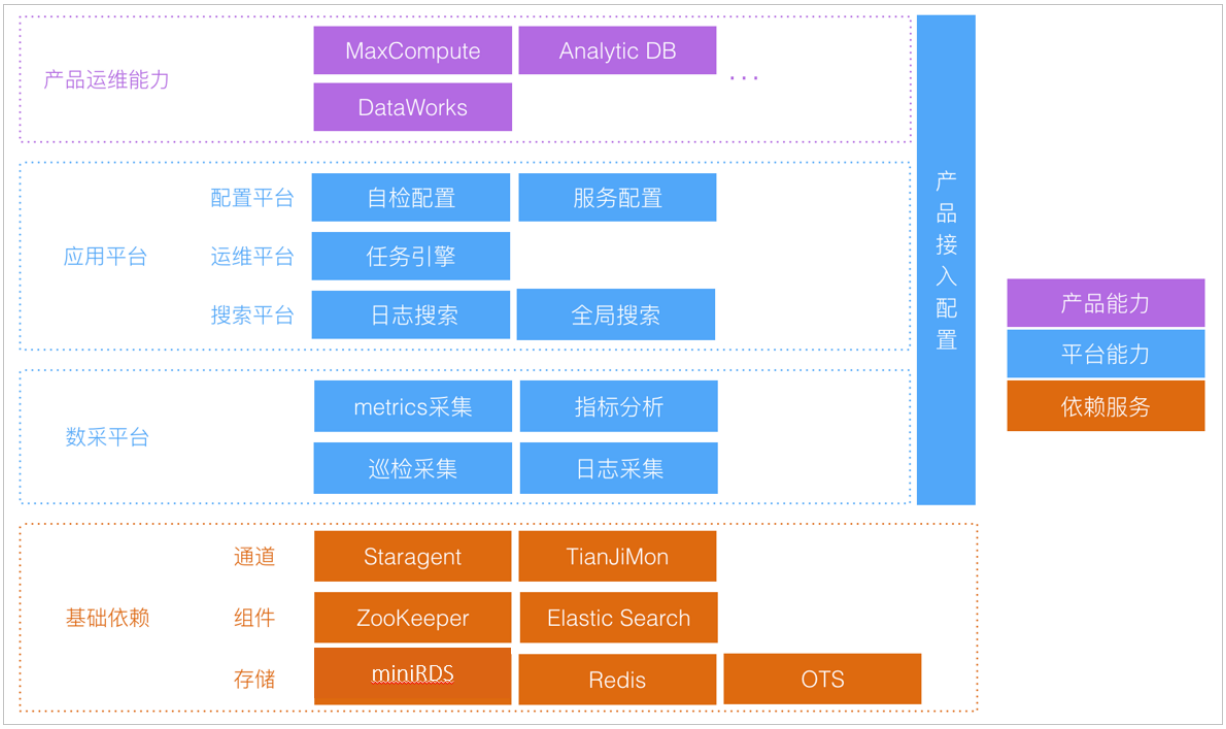
大数据管家（Big Data Manager)是一款基于层次化服务管控方案的大数据产品管控平台，管控着包括MaxCompute、DataWorks、Analytic DB以及众多数据产品，并为这些产品提供服务监控、日常自检、日志搜索、运维、配置以及服务特性运维功能，维护产品稳定性。

10.2 产品架构

大数据管家采用微服务设计理念，在此之上提供数据整合、接口整合以及功能整合的统一整合平台，暴露统一标准的服务接口。基于这样的设计理念，在大数据管家上的不同产品的页面操作和运维体验完全一致，减轻现场运维学习成本，降低运维风险。

整个系统主要由基础依赖、数采平台、应用平台、以及产品运维能力组成，如图 *10-1:* 产品架构所示。

图 10-1: 产品架构



10.2.1 基础依赖

大数据管家采用了集团统一的通道服务staragent和TianJiMon来完成远程命令和远程数据采集指令，通过zookeeper来协调主从服务，保证服务可用性。通过Elastic Search来完成在大数据管家管理下的所有数据的索引，为上层提供各类搜索服务打好数据基础。在存储方面，大数据管家主要使用miniRDS来存储元数据，使用redis作为缓存服务，使用OTS来存储大量的自检采集信息，提升吞吐量。

10.2.2 数采平台

依赖TianJiMon，大数据管家开发了针对实时运维数据采集的采集脚本和针对服务状态采集的巡检脚本，通过回收这些脚本采集的数据，可以快速了解服务整体运行数据和状态，通过一定规则的聚合、筛选和甄别发现有价值信息。日志采集则是依赖staragent完成实时采集任务，让您可以实时分布式抓取日志信息。

10.2.3 应用平台

应用平台根据底层提供的能力以及数据采集的数据信息，构筑出多个提供上层服务能力的平台，这里主要包含配置平台、运维平台以及搜索平台。

- 配置平台可以方便的让您了解指定服务的所有配置信息，主要包含自检的配置和服务本身的配置，并且这些配置可以在这里修改提交。
- 运维平台主要提供运维的能力，并且可以智能识别您当前所在的服务而动态填写参数。
- 搜索平台主要提供了全局搜索和日志搜索能力。
- 全局搜索是针对所有数据的一个通用搜索，根据不同的搜索结果产生定制化内容，如图 *10-2: 全局搜索*所示。
- 日志搜索则提供了针对指定服务的指定日志路径的实时日志搜索能力，如图 *10-3: 日志搜索*所示。

图 10-2: 全局搜索

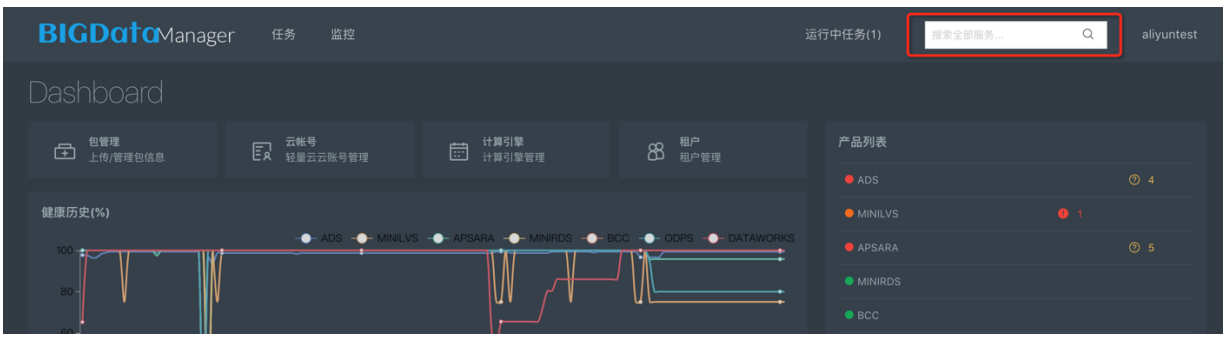
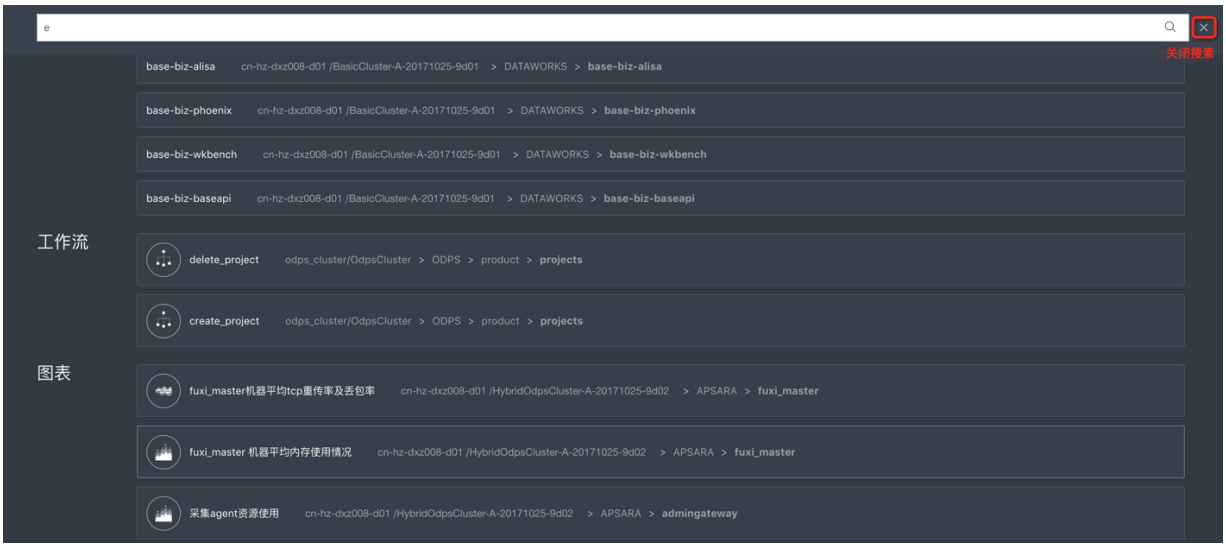


图 10-3: 日志搜索



10.2.4 产品运维能力

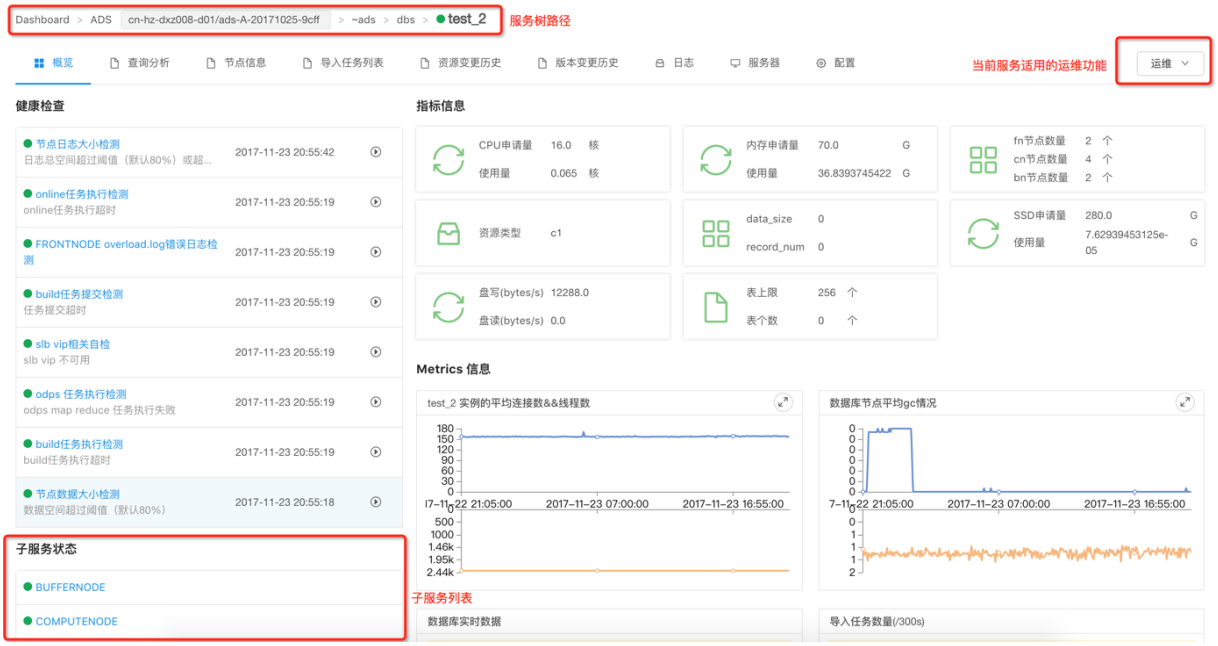
这三个平台并不意味着就是所有大数据管家上提供的产品运维能力，因为部分跟产品有关的定制化运维功能则依托在每一个产品运维能力本身上。他们都会深度的与具体需要运维的产品融合和定制，达到一致性和特殊性的统一。例如AnalyticDB 会在日志平台的能力上，再提供更高层次封装的真对其服务的查询分析，并结构化展示分析数据，让您可以快速了解日志信息。这些定制都是透明的，您在具体运维场景上会顺利地了解这些定制内容。

10.3 功能特性

10.3.1 层次化服务管控

大数据管家主要使用层次化服务树来分解应用，也同时引导您了解被运维产品的特性，如图 *10-4: 服务管控*所示。

图 10-4: 服务管控



层次化服务树主要是对被运维应用的一种树形产品模块梳理，这种梳理可能面向服务组件梳理，也可能面向业务场景梳理，梳理出来后，每一个层次上的节点我们都称之为服务。

在这个树里您可以轻松的发现这个服务的父服务，以及它所依赖的子服务。对于服务本身的所有操作，都可以在这个服务的页面上找到。

服务树主要有以下几个特性。

- 产品树里面的服务可配置并实时动态更新。
- 支持跨产品依赖。
- 飞天、机器信息统一配置并快速接入。
- 按照产品树做信息展示和信息汇聚。
- 按照产品树做健康检查及其状态汇聚。

10.3.2 自我管控

大数据管家有自我管控能力，可以在应用平台层或底层依赖服务出现问题时显示问题定位和原因，方便您快速恢复大数据管家。

10.3.3 管控服务列表

大数据管家下管理的产品及它们的别名如下表所示。

产品名称	别名
MaxCompute	ODPS
Analytic DB	ADS
DataV	-
DataWorks	Base
DTBoost	-
IPlus	I+
StreamCompute	Galaxy
PAI	-
Quick BI	-
Apsara	-
BCC	大数据管家

10.4 故障处理

常见故障处理

- **登录故障**

  如出现无法登录的情况，请先清理浏览器的缓存、cookie，重新尝试登录。

  如果登录框提示登录异常，请根据提示异常检查以下内容。

  - 是否密码不对
  - 是否账号锁定
  - 是否账号被关停

- **其他异常**

  请寻找技术支持。

11 关系网络分析

11.1 产品概述

阿里云关系网路分析软件（简称I+）是阿里云推出的一款基于关系网络的海量数据智能可视化研判平台。

产品面向公安、工商、税务、海关、银行、保险、互联网金融等领域的大数据情报分析，为案件分析研判、反洗钱、反欺诈、反腐反贪、关联交易等调查分析提供强力支撑，帮助分析人员洞察数据中的关键信息，快速、智能的寻找破案线索及有价值的情报。

11.2 产品架构

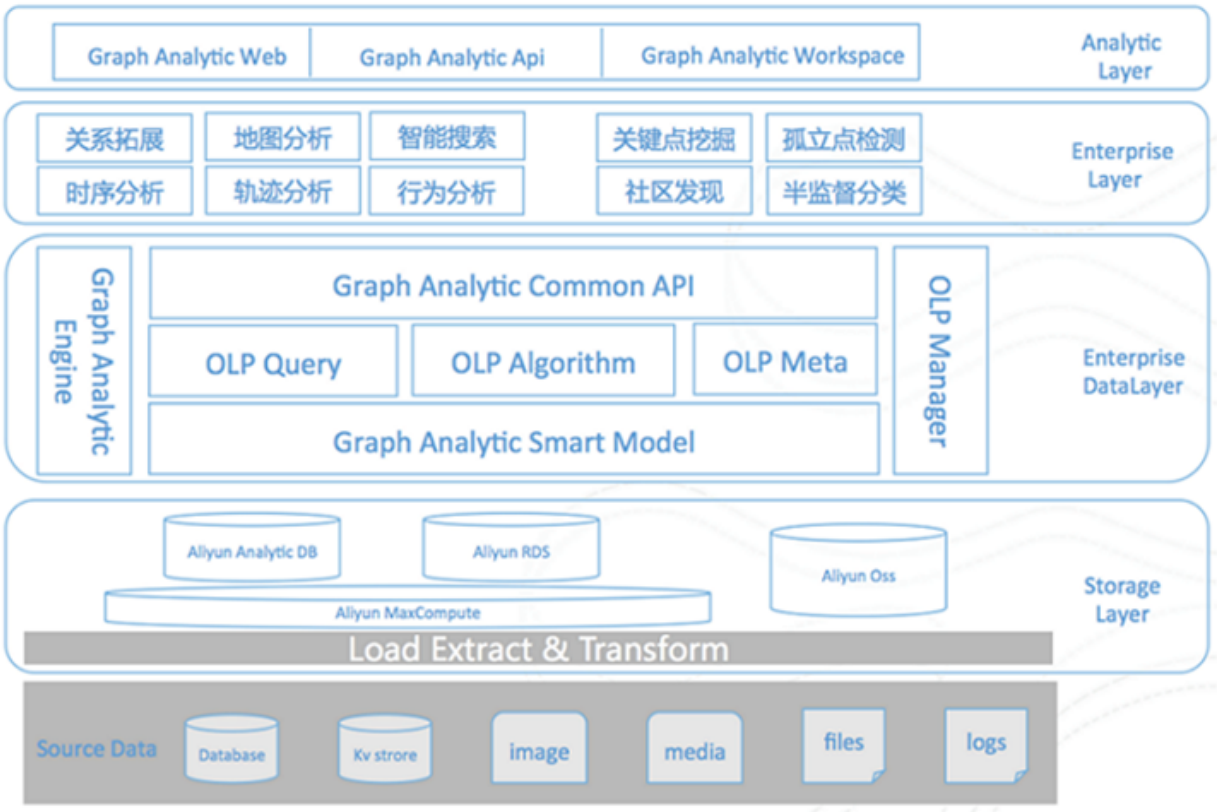
阿里云关系网路分析软件（Graph Analytic）（简称I + ）采用组件化、服务化设计理念，多层次体系架构。

数据存储计算平台建立在阿里云自主研发的大数据基础服务平台（数加平台）上，支持 PB/EB 级别数据的存储和计算，具有强大的数据整合、处理、分析、计算能力。

I+ 的系统架构如图 *11-1:* 系统架构图所示。

整个系统分为存储计算层、数据服务层、业务应用层、分析展现层。

图 11-1: 系统架构图



- **存储计算层**：基于阿里云大数据平台，支持多种开放数据源。计算平台分为离线和在线，离线计算为 MaxCompute，实现数据的整合、处理，在线计算实现数据的实时计算，包括分析型数据库 AnalyticDB、图数据库（BigGraph）、流计算（StreamCompute）。
- **数据服务层**：按照关系域、关系类型、关系事项抽象出的“实体 - 属性 - 关系”模型，提取自然对象关系、社会对象关系、空间对象关系，进行业务关系逻辑建模，通过逻辑业务定义整合异域多源的数据，支持逻辑模型的灵活管理和维护。数据服务引擎为业务应用层提供统一的业务逻辑查询语言，执行各种复杂的关系网络查询、算法分析。
- **业务应用层**：将关联网络、时空网络、搜索网络、信息立方、智能研判、协作共享、动态建模等多业务业务应用封装成API接口，提供给分析层调用。
- **应用分析层**：提供多元智能可视化交互分析界面，支持多种终端。提供外部 API 接口及可视化组件服务，支持第三方系统的接入。

11.3 功能特性

11.3.1 关系网络

关系网络是I + 研判分析平台的核心模块，所有实体之间的关系拓扑，业务计算，可视化布局，以及图交互操作在其中。同时有三大辅助分析组件用户空间、时序分析和信息立方补充关系网络研判，使之可以涵盖大部分研判业务场景。

11.3.2 时空网络

时空网络引入时间、空间维度，将实体、关联等数据和 GIS 地图结合，发掘出空间关联关系，并利用机器学习，智能计算同轨伴随、空间活动等信息。

11.3.3 搜索网络

搜索网络提供对象信息检索的功能，用户在分析过程中逐步递进拓展信息，从线索关键词开始细化分析，如**电话号码**：**138*****001**，**姓名**：**张三**，**住址**：**杭州市西湖区文三路**号**等。搜索提供检索工具帮助用户快速定位信息，同时作为关系网络和时空分析的入口，可以将检索对象信息引入到**关系网络**和**时空网络**继续拓展分析。

11.3.4 信息立方

产品提供图形、表格等多视角信息呈现方式，包括行为分析、时序分析、行为明细、网络统计、属性分布统计等，帮助用户进行多维信息洞察。

11.3.5 智能研判

智能研判是I + 平台上旨在研判过程中为业务提供智能化线索的算法应用。这些算法是I + 算法同学深入研判领域如公安、反恐以及税务中沉淀出的业务智能。目前I + 已经沉淀的伴随分析、涉恐指数、吸毒指数等应用，在多个项目中取得重大成果。

11.3.6 动态建模

复杂时空大数据往往以复杂关联网络形式存在，基于本体论和语义网技术，产品中设计实现大数据背景下通用领域抽象数据模型。用对象、关系的链接抽象的方式来组织刻画数据，将数据析解成对象（Object）、属性（Property）以及对象间的关联关系（Link），形成OLP数据模型。OLP数据模型支持用户快速组织整合数据，同时支持业务模型灵活扩展。

11.4 产品优势

11.4.1 超大规模计算及存储

I+数据存储计算平台建立在阿里云自主研发的大数据基础服务平台（数加平台）上，支持PB / EB级别数据规模的处理。计算存储平台包括大数据计算服务（MaxCompute）、分析型数据库（AnalyticDB）、分布式图计算（BigGraph）等。

大数据计算服务（MaxCompute）

阿里云大数据服务 MaxCompute，单个集群的规模可达5000台，并且具备跨机房的线性扩展能力，轻松处理海量数据。离线调度支持百万级任务量，实时监控告警。

核心指标：

- 万亿级数据 join，百万级并发 job，作业 I/O 可达 PB 级/天。
- 具备跨集群（机房）数据共享能力，支持万级别的集群数，扩容不受限制。
- 提供功能强大易用的 SQL、MR 引擎，兼容大部分标准 SQL 语法。
- MaxCompute（原ODPS）采用三重备份、读写请求鉴权、应用沙箱、系统沙箱等。多层次数据存储和访问安全机制保护用户的数据不丢失、不泄露、不被窃取。

分析型数据库（AnalyticDB）

阿里云分析型数据库（Analytic DB）是基于 MPP 架构并融合了分布式检索技术的分布式实时计算系统。

核心指标：

- 拥有快速处 理迁移级别海量数据的能力，使得数据分析中使用的数据可以不再是抽样的，而是业务系统中产生的全量数据，使得数据分析的结果具有最大的代表性。
- 采用分布式计算技术，拥有强大的实时计算能力，通常可以在数百毫秒内完成百亿级的数据计算，使得使用者可以根据自己的想法在海量数据中自由的进行 探索。
- 支撑较高并发查询量，并且通过动态的多副本数据存储计算技术来保证较高的系统可用性。

分布式图计算软件（BigGraph）

阿里云分布式图计算软件（BigGraph）是一款低延时高可用的分布式图计算产品，适用于大数据上的交互式图分析场景。

核心指标：

- 分布式存储

- 分布式计算
- 兼容 Gremlin 图遍历语言
- 基于多备份的高可用机制

11.4.2 跨计算数据整合建模，灵活高效部署

I + 平台以逻辑模型的建立替代耗时耗力的传统数据仓库物理模型，为分布在多个计算存储资源上的明细数据建立统一的“实体-关系-属性”的逻辑模型。用户在业务数据理解和梳理后，通过I+管理后台轻松配置定义业务逻辑模型，配置逻辑模型和实际物理数据存储的映射关系，以及应用场景的参数即可以完成业务分析功能的建模和定义。逻辑模型支持根据数据源的变动或业务变动，进行灵活的修改和实时部署生效。

以目前I+实际落地的项目来看，在充分理解业务数据的基础上，通常在1-2天就能完成系统部署应用上线使用。

11.4.3 智能算法组件集成，挖掘数据价值

I + 平台关系网络引擎为关系型数据的挖掘提供了多种针对业务优化的关系网络模型和算法运行计算，平台结合经典战法，融入机器学习、业务算法，实现智能分析，如犯罪系数、伴随分析、骨干分析、路径分析等，帮助用户快速实现对关系网络数据的复杂挖掘。

11.4.4 智能可视化交互，提升用户体验

I + 可视化交互界面结合关系网络、地图分析、信息立方、时序行为面板等多个维度的信息交互提供给分析用户，方便用户从网络、时间、空间三个维度视角来探查分析。同时提供各种常用应用工具的操作，支持用户常用操作习惯，提升用户体验。

11.4.5 高度参数配置化，实现灵活的项目定制

I+平台支持包括数据源、业务模型，样式图标，用户角色权限，技术参数、系统参数等基本上所有可配置参数的配置化管理。用户可以根据项目的实际情况在管理控制台灵活自定义配置，满足具体项目的实际业务需求。

11.5 产品价值

目前阿里云关系网路分析软件（简称I + ）已经作为亮点应用参与了多个国家级重点项目，涉及包括公安、金融、国税、保险、互联网金融等多个行业多个领域，尤其在安保、反恐、反洗钱、税务欺诈等细分领域的应用让客户耳目一新，业务价值得到深度考验和认可。

下面列举一些典型行业的典型应用案例，说明阿里云关系网络分析软件在真实场景中如何通过交互式的所见及所得的方式，让用户在复杂数据环境中理清数据之间的关联和逻辑关系，并通过直观和友好的用户界面展现给用户。

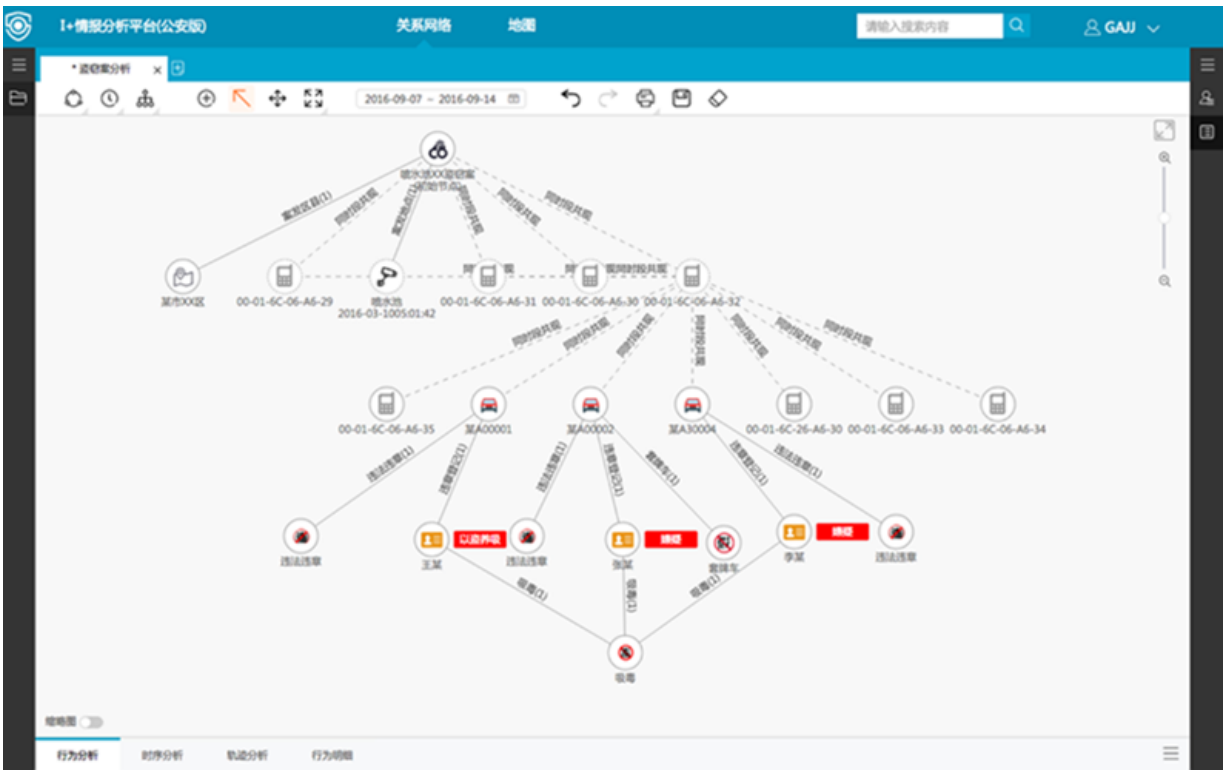
说明：
案例截图来源于真实应用系统，相关的案例敏感数据已做脱敏处理。

11.5.1 公安行业应用

此案例描述的是公安办案人员如何通过阿里云关系网路分析软件（简称I + ）快速侦破**以盗养吸**案件。

2016年以来，某地连续发生店铺被盗案件，店铺柜台、抽屉遭到不同程度的破坏，现金遗失、被盗。办案人员根据掌握的虚拟账号位置信息、车辆轨迹信息，结合时间、空间元素，利用阿里云关系网路分析软件（简称I + ）对多起案件进行串并分析，如图 *11-2: 盗窃案分析*所示，发现在多个案发现场、案发时段均出现相同的虚拟账号，根据已掌握的信息中未能直接分析得到虚拟账号对应的人员，但办案人员结合车辆轨迹、虚拟账号移动轨迹进行时空伴随分析，关联挖掘虚拟账号对应的车辆，进一步反查发现车主为吸毒前科人员，具有较大作案嫌疑，为整个案件的侦破提供了重要线索。

图 11-2: 盗窃案分析



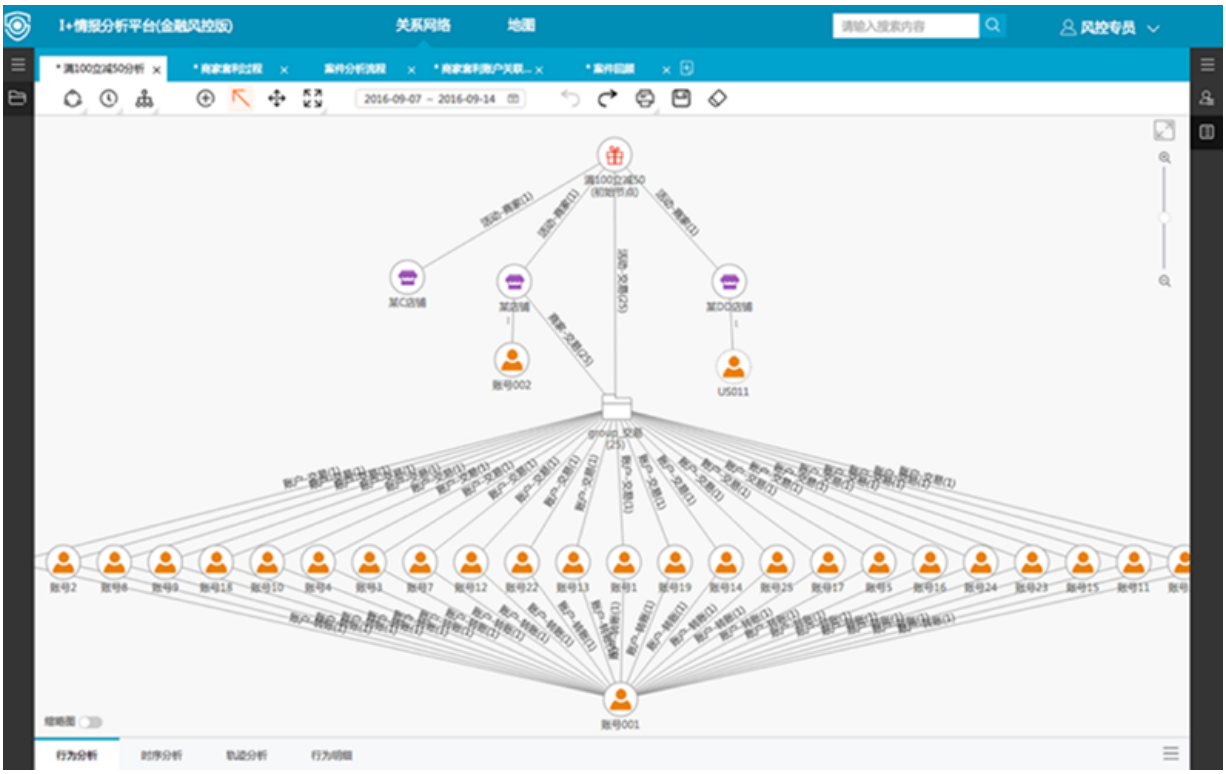
11.5.2 金融行业应用

此案例描述的是互联网金融风控专员如何通过阿里云关系网路分析软件（简称I + ）进行**营销反作弊**的分析。

某互联网企业投放大额资金在线下店铺开展买100立减50的纳新营销活动，部分店铺企图套取营销资源，通过事先批量注册小号并在活动前进行转账激活，活动当天在店铺购买物品产生虚假交易，骗取营销资源并获利。

该互联网企业综合利用转账、位置、设备、环境等信息，构建可疑关系网络，通过I+情报分析平台进行快速定位分析，如图 *11-3:* 营销反作弊分析所示，最终追回被套利资源，避免了营销资源的浪费，同时对相应的环境及账号进行布控，对该店铺进行相应的处罚。

图 11-3: 营销反作弊分析



11.5.3 税务行业应用

此案例描述的是税务分析人员如何通过阿里云关系网路分析软件（简称I + ）进行**出口骗税**的分析。

某外贸综合服务平台为中小企业提供专业、低成本的通关、外汇、退税以及配套物流和金融服务，但是部分企业存在投机取巧、造假骗税的情况，骗税金额不断攀升，对平台的外贸资质带来重大负面影响。

分析人员利用I+情报分析平台的关联反查、群体分析、骨干分析、共同邻居、信息立方以及行为分析等功能，如图 *11-4:* 出口骗税分析所示，从被举报的企业入手，查获了一个涉及13家企业的大规模骗税团伙，该团伙在平台上注册不同角色，内部间虚开发票、作假交易，向贸易平台申报退税金额达到上亿人民币，实地看厂调查后，对该团伙实施了相应的处罚，并追回损失。

图 11-4: 出口骗税分析