搭建私有化DataWorks平台

2019年10月20日 分类：科技

本文共6617个字，阅读需要17分钟。

以下是起点数据中台为您报道关于【搭建私有化DataWorks平台】的具体情况和说明，并在快啵www.kuaiboo.net网起点数据中台以图文形式为您慢慢道来，本文热门关注焦点《数据, 数据同步, 平台》。

前言

搭建私有化阿里云DataWorks平台需要三个大的维度的功能：

数据同步

数据开发

数据质量

数据开发Part 1已经讲完了，本文会深入讲解数据同步，数据质量会留在下一篇文章深入讲解。

数据同步的核心

数据同步是企业拥抱数据中台真正落地的第一步，这一步至关重要。这步核心要解决的事情是：

确定围绕着本期数据中台规划需要达到的目标，围绕着这个目标我们需要明确我们要支持哪些Source源（就是数据来源）：

存在于传统关系型数据库中都有哪些库、表、字段？

这些库、表、字段具体的含义？这个需要我们更好的理清楚，这样便于数据开发做开发。

有哪些数据是已经在企业的各种关系型数据库（包括MySQL、Oracle、PG、DB2、Sybase等）中，可以直接使用的？

有哪些是需要从第三方渠道爬取或者是购买的？这些数据最终会保存在哪里？是直接放在关系型数据库中，还是直接采集后发送到类似于Kafka这样的消息队列中？

有哪些数据是通过IOT方式获取到的？是直接流入到时序数据库中直接做处理，还是先流入Kafka这样的消息队列，以备后续使用？

数据是原材料，是必不可少并且又不能缺失的。

需要哪些维度的数据?

这些数据都分布在哪里?

这些数据如何更好的盘点?资产盘点（或者说数据盘点）是一个很大的范畴，理论上是咨询公司对外提供数据中台构建解决方案的过程中帮助企业更好的提供解决方案的第一步；也是数据中台技术输出公司对外提供技术输出的第一步！这个是术、方法论层面的，后续有机会再细讲，本篇就不深入展开了。对于我们，是需要理清:

围绕着数据的开发过程，需要明确数据的Sink源（就是数据同步环节数据去处）：

如果只需要离线数仓搭建，数据直接同步到HDFS、Hive表就可以了。

如果还需要实时数仓搭建，那就必须要把数据Sink到Kafka中，供Flink这样的实时引擎消费、加工。

是只需要离线数仓还是也需要实时数仓?

小结

总结一下，数据同步抛开术层面（就是方法论层面，这块主要体现在资产盘点这个小环节，当然不是每个公司都需要），我们需要明确:

需要支持的Source源:

关系型数据库：Mysql、Oracle、Sybase、PG、DB2?

消息队列：Kafka?

需要支持的Sink源:

HDFS、Hive（离线场景）

Kafka（实时场景）

Hbase?（这个不常见，可以忽略）

ES?（这个不常见，可以忽略）

数据同步需求明确

数据同步这个环节，从满足大部分企业需要而言，需要做以下支撑:

必须功能:

数据同步这个环节一定是可视化配置的（包括数据源选取、目标源选取、同步方式选取）。这些最好是同一套操作方式，通过Web端可视化配置、提交就可以一键数据同步过程，把底层具体的细节尽可能屏蔽掉。

数据同步这个过程有基本的监控:

包括有基本的流量监控;

有成功或者失败监控;

有使用过程资源消耗（包括CPU、内存、网络IO）等环节监控;

全量同步

增量同步

Hive

HDFS

Kafka

Mysql

Oracle

PG

支持常用关系型数据库数据源：

支持常见目标源：

支持不同同步方式，实时&离线数仓需求：

易用：

重要功能

数据实时脱敏；

数据源schema变更感知：当源端发生schema变更时，能自动感知schema变化；

多租户；

权限控制；

报警支持；

可靠多路消息订阅分发：根据同一套数据源，根据离线&实时数仓需求，支持多路消息订阅以及分发；

以上必须功能是我们做好数据同步这个环节必须要支持的；重要功能是非常重要，但是不一定要完全都支持。但是尽量要支持，这样才能更好的满足各个公司、各种业务的需求。

技术选型

针对我们需要支持的功能细分，底层可选的引擎根据功能如下：全量数据同步：DataX是最好的选择！如下是节选DataX github介绍：DataX 是阿里巴巴集团内被广泛使用的离线数据同步工具/平台，实现包括 MySQL、Oracle、SqlServer、Postgre、HDFS、Hive、ADS、HBase、TableStore(OTS)、MaxCompute(ODPS)、DRDS 等各种异构数据源之间高效的数据同步功能。

增量数据同步：

Canal：MySQL binlog 同步到 Kafka

Ogg：Oracle 同步到 Kafka

Flume：支持处理各种类型、各种格式的日志数据，包括Avro Source、Exce Source、Spooling Directory Source、NetCat Source、Syslog Sources、Thrift Source、Sequence Generator Source、HTTP Source、Kafka Source等数据源

其他各种增量同步工具

所以，从底层技术引擎来看，如果是单纯支持常用数据源Sink到Kafka、Hive、HDFS，支持全量和增量同步类型，Flume + DataX就能够满足基本要求了。

不过这块涉及几个问题：

不够易用，使用成本很高：

使用方法不统一。需要根据增量、全量自己封装底层的插件：Flume或者DataX，调用它们自身的命令。

不能可视化操作。

没办法做流量监控。

其他的上文提及的重要功能都没办法直接支持，需要做大量定制化的功能才可以。

所以，这块不能直接使用DataX、Flume等插件，而是需要使用已经做了大量封装，在功能、易用性、可运维性（包括监控&报警完善）、多租户隔离等等都做了大量支持和完善。

可选的方案

其实满足这块的系统还是蛮少的，不过还是有，可选的有如下几个：DataLink：优车集团大数据团队开源的。DataLink是一个满足各种异构数据源之间的实时增量同步，分布式、可扩展的数据交换平台。DBus：宜信大数据团队开源的。专注于数据的收集及实时数据流计算，通过简单灵活的配置，以无侵入的方式对源端数据进行采集，采用高可用的流式计算框架，对公司各个IT系统在业务流程中产生的数据进行汇聚，经过转换处理后成为统一JSON的数据格式（UMS），提供给不同数据使用方订阅和消费，充当数仓平台、大数据分析平台、实时报表和实时营销等业务的数据源。

DataLink VS DBus

DataLink相比DBus更轻量级些，依赖的组件更少，只依赖Zookeeper、MySQL这两个组件，所以非常轻量级。但是这货存在如下几个问题：

功能太简陋

易用性不是很好

稳定性、可扩展不是很好：DataLink它并没有使用DataX、Flume这样的组件，而是完全自己写的同步的逻辑。自己写倒没问题，但是会让人担心它的稳定性以及可扩展性，后续如何增新的数据源，新的同步模式还是自己重新开始撸代码，成本太高了。

综合评估了下，DataLink就不做考虑了。所以DBus是唯一的选择，而且确实是非常好的选择。

数据通道-DBus

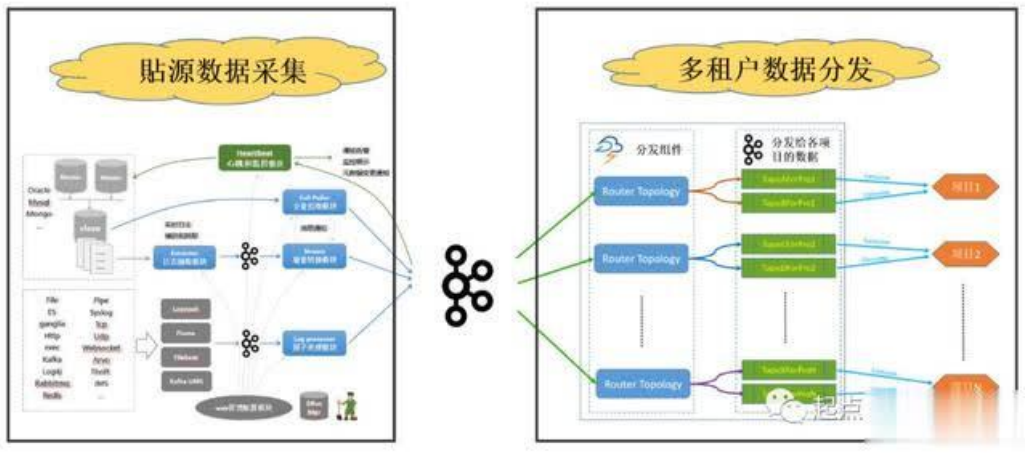
DBus确实很强大，让人又爱又恨，先说让人爱的地方，再说恨吧！

如下是DBus系统架构和工作原理，为了方便读者更好的了解DBus，节选了DBus官网的一些介绍，让大家有个基本的认识。

DBus介绍

系统架构和工作原理

DBUS主要分为两个部分：贴源数据采集和多租户数据分发。两个部分之间以Kafka为媒介进行衔接。无多租户资源、数据隔离需求的用户，可以直接消费源端数据采集这一级输出到kafka的数据，无需再配置多租户数据分发。



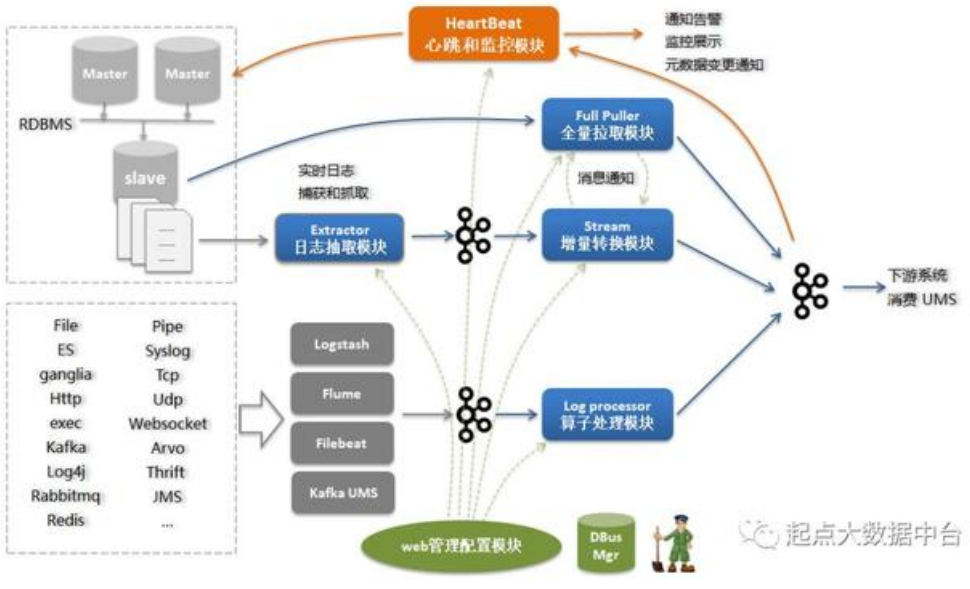
1 DBUS源端数据采集

DBUS源端数据采集大体来说分为2部分：

读取RDBMS增量日志的方式来 实时获取增量数据日志，并支持全量拉取；

基于logtash，flume，filebeat等抓取工具来实时获得数据，以可视化的方式对数据进行结构化输出；

以下为具体实现原理



主要模块如下：

日志抓取模块：从RDBMS的备库中读取增量日志，并实时同步到kafka中；

增量转换模块：将增量数据实时转换为UMS数据，处理schema变更，脱敏等；

全量抽取程序：将全量数据从RDBMS备库拉取并转换为UMS数据；

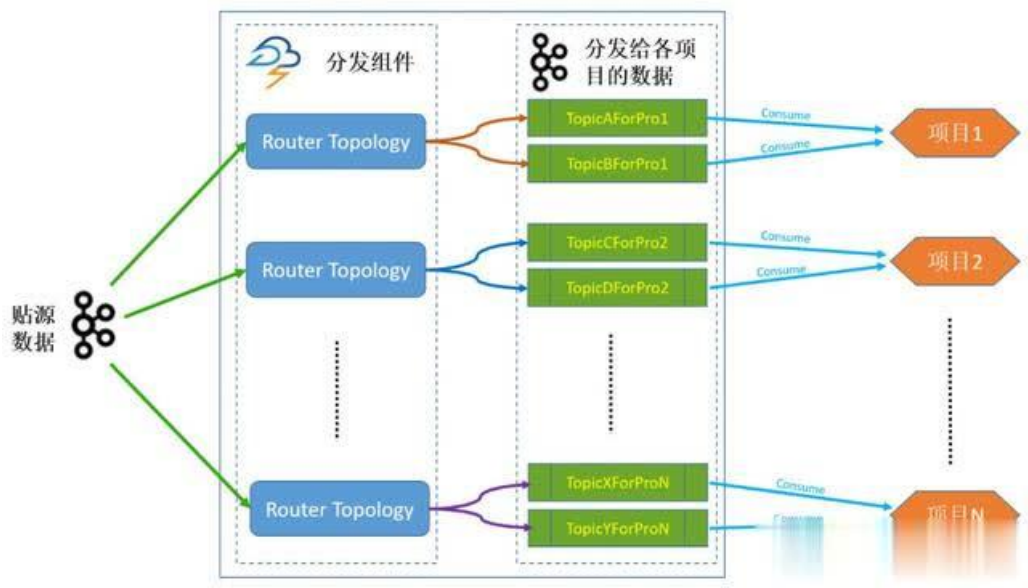
日志算子处理模块：将来自不同抓取端的日志数据按照算子规则进行结构化处理；

心跳监控模块：对于RDMS类源，定时向源端发送心跳数据，并在末端进行监控，发送预警通知；对于日志类，直接在末端监控预警。

web管理模块：管理所有相关模块。

2 多租户数据分发

对于不同租户对不同源端数据有不同访问权限、脱敏需求的情形，需要引入Router分发模块，将源端贴源数据，根据配置好的权限、用户有权获取的源端表、不同脱敏规则等，分发到分配给租户的Topic。这一级的引入，在DBUS管理系统中，涉及到用户管理、Sink管理、资源分配、脱敏配置等。不同项目消费分配给他的topic。

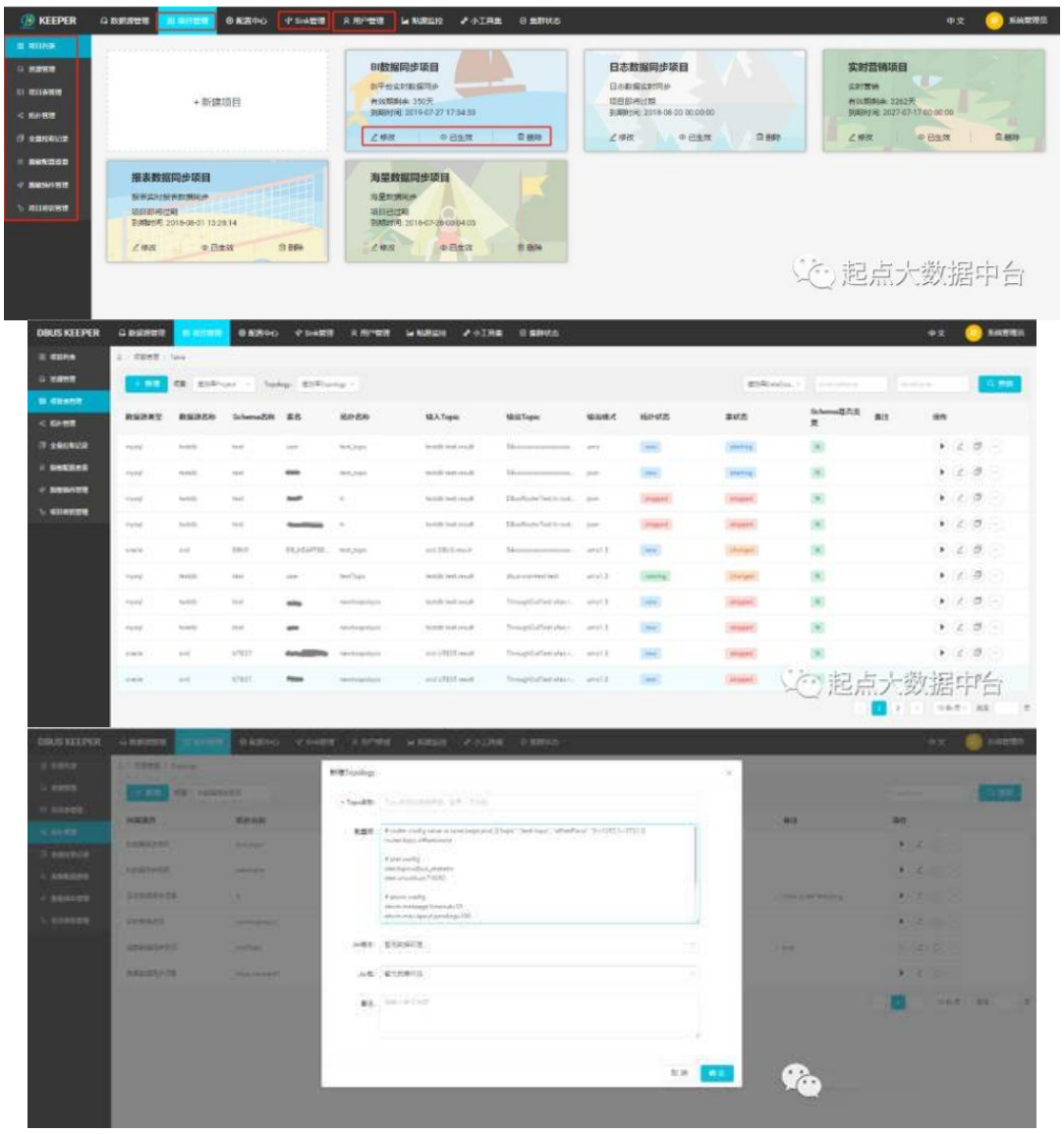


主要功能：

无侵入方式接入多种数据源：业务系统无需任何修改，以无侵入性读取数据库系统的日志获得增量数据实时变化。目前RDBMS支持mysql，oracle数据源（Oracle数据源请参考Oracle相关协议）， 日志方面支持基于logstash，flume和filebeat的多种数据日志抽取方案。

海量数据实时传输：使用基于Storm的流式计算框架，秒级延时，整体无单点保证高可用性。

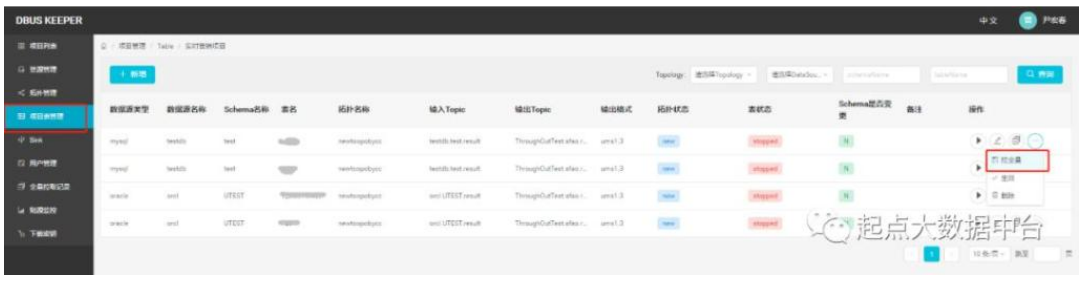
多租户支持：提供用户管理、资源分配、Topology管理、租户表管理等丰富的功能，可根据需求，为不同租户分配不同的源端表数据访问权限，应用不同的脱敏规则，从而实现多租户资源隔离、差异化数据安全。



感知源端schema变更：当源端发生schema变更时，能自动感知schema变化，调整UMS版本号，并通过Kafka消息和邮件通知下游

数据实时脱敏：可根据需求对指定列数据进行实时脱敏。脱敏策略包括：直接替换、MD5、murmur等脱敏算法，脱敏加盐，正则表达式替换等。支持用户开发jar包实现DBUS未覆盖的个性化脱敏策略。

初始化加载：支持高效的初始化加载和重新加载，支持任意指定输出topic，灵活应对客户需求。



统一标准化消息传输协议：使用统一的UMS(JSON格式)消息schema格式输出便于消费，提供数据线级ums_id保证数据顺序性,输出insert,Update(before/after),Delete event数据。

```
{
    "protocol": {
        "type": "data_increment_data",
        "version": "1.3"
    },
    "schema": {
        "namespace": "mysql.testdb.schema1.table1.2.0.0",
        "fields": [
            {"encoded": false, "name": "ums_id_", "nullable": false, "type": "long"},
            {"encoded": false, "name": "ums_ts_", "nullable": false, "type": "datetime"},
            {"encoded": false, "name": "ums_op_", "nullable": false, "type": "string"},
            {"encoded": false, "name": "ums_uid_","nullable": false, "type": "string"},
            {"encoded": false, "name": "id", "nullable": true, "type": "long"},
            {"encoded": false, "name": "customerName", "type": "string"},
            {"encoded": false, "name": "customerType", "nullable": true, "type": "int"},
            {"encoded": false, "name": "signUpState", "nullable": true, "type": "int"},
            {"encoded": false, "name": "createTime", "nullable": false, "type": "datetime"}
        ],
        "batchId": 2
    },
    "payload": [
        {"tuple": ["0025390000550881764", "2017-08-24 17:19:12.000", "i", "4345133", "101", "user1", 1, "OK", "2017-08-24 17:19:12"]},
        {"tuple": ["0025390000550900537", "2017-08-24 17:19:12.103", "u", "4345133", "102", "user2", 3, "failed", "2017-08-24 17:19:12"]},
        {"tuple": ["0025390000550918096", "2017-08-24 17:19:12.532", "d", "4345133", "103", "user3", 4, "OK", "2017-08-24 17:19:12"]}
    ]
}
```

可靠多路消息订阅分发：使用Kafka存储和传递消息保证可靠性和便捷的多用户订阅

支持分区表/系列表数据汇集：支持分区表的数据汇集到一个“逻辑表”。也可将用户自定义的系列表数据汇集到一个“逻辑表”。例：

实时监控&预警：可视化监控系统能随时查看各数据线实时流量和延时状况；当数据线发生异常时，根据配置策略自动发邮件或短信通知相关负责人

DBus优点

不得不说，看到DBus这个项目难掩我内心的狂喜，太完美了。完全满足了我对数据同步的一切想象，从功能、架构、可扩展性、易用这些方面再过几年都不过时，而且完全能够满足中大型互联网公司的所有需求，对于传统企业或中小型公司完全不在话下。太完美了，优点太多，就不赘述，感兴趣的可以深入研究下这个项目。

DBus缺点

任何事物都有两面性，有强的必有其弱，有时候强的事物弱的地方就是因为其强。就如DBus来说吧，它强在功能的强大、可扩展性非常好、架构有足够的前瞻，由于这些强，使得它存在以下的问题：

依赖外部组件非常多，它依赖：

Canal

Zookeeper

Kafka

Storm

Mysql

Influxdb

Grafana

使得部署这样一套系统成为了不小的挑战，而且这些节点的维护以及运维成本都挺高的，部署需要的资源也蛮多的。当然，如果从业务的前瞻性，系统未来的可扩展而言，功能强大些也没有坏处。只是前期稍微忍受下其部署麻烦，占用资源稍微多些的缺点就好了。

功能并不完整，需要结合Wormhole一起使用才能完全搞定数据同步这件事情：这个也是DBus另外一个由于它的强导致的缺点。因为它的目标是做一个通用、完善的系统，它建立了一个很强大的抽象，把数据流动抽象成管道，管道这一侧是各种传统类型的数据库源，管道目标测是自定义的UMS格式描述。由于建立了这个抽象，使得数据并没有Sink到我们想要的目标

源，比如Hive、HDFS等。不过好在有Wormhole这样的系统解决了这个问题，使得它们配合可以完美解决我们一切问题。

DBus + Wormhole赋能数据同步

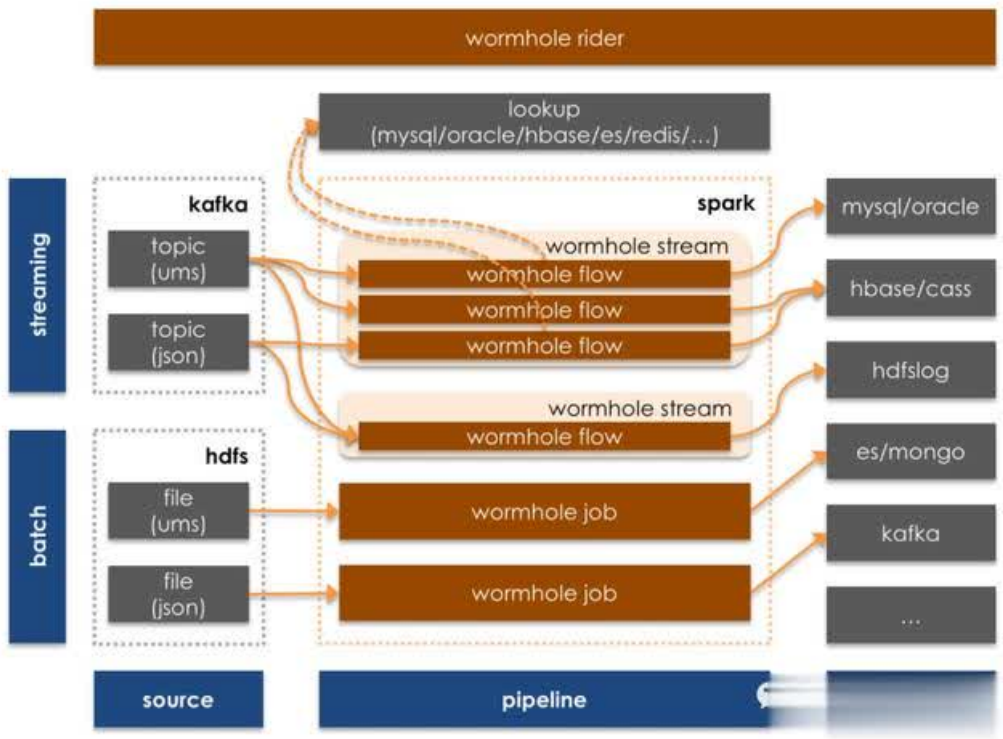
DBus上节已经讲过，这块就不再赘述了。从数据流动的角度来看，DBus把数据做一些处理后，流到Kafka这样的消息队列中，用自定义的UMS做描述。

如果把数据Sink到我们想要的目标数据源：比如Hive、HDFS等，需要借助Wormhole的帮助。Wormhole其实也是一个非常强大的系统，而且部署起来非常容易，易用性，功能完善性，稳定性都非常不错。

这块稍微介绍下Wormhole吧，详细的内容可以查看Wormhole官网：

Wormhole概述

架构：



设计理念：

统一 DAG 高阶分形抽象

构建由 Source DataSys，Kafka Topic，Spark Stream（Flink Stream），Sink DataSys 组成的物理 DAG

每个物理 DAG 里可以并行处理多个由 Source Namespace，Flow，Sink Namespace 组成的逻辑 DAG

每个 Flow 本身是典型的 Spark RDD DAG

统一通用流消息 UMS 协议抽象

UMS 是 Wormhole 定义的流消息协议规范

UMS 试图抽象统一所有结构化消息

UMS 自身携带结构化数据 Schema 信息

Wh4 支持用户自定义半结构化 JSON 格式

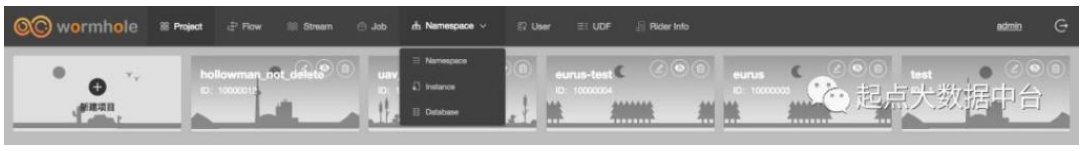
统一数据逻辑表命名空间 Namespace 抽象

Namespace 唯一定位所有数据存储所有结构化逻辑表

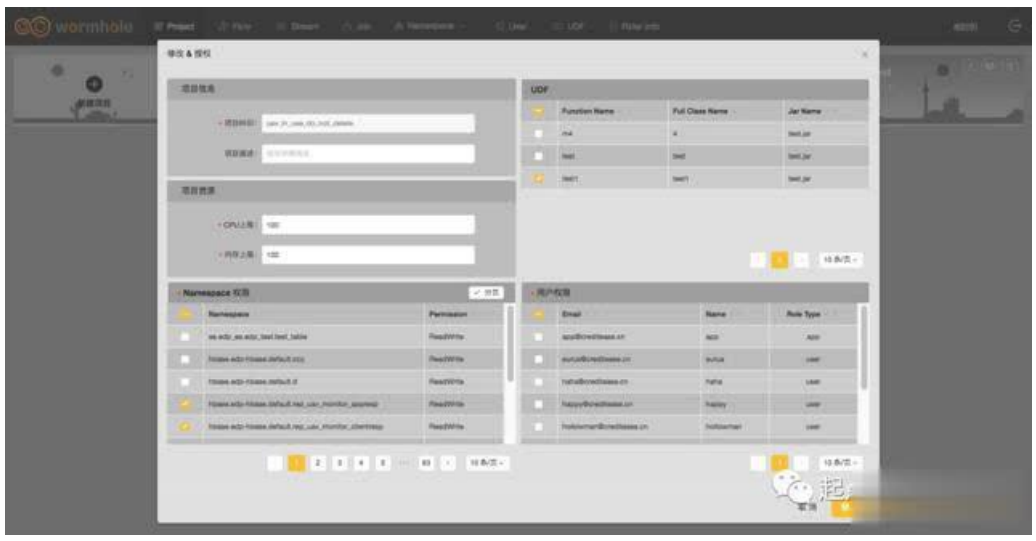
[Data System].[Instance].[Database].[Table].[Table Version].[Database Partition].[Table Partition]

实践：

Admin 可以创建 Project/Namespace/User/UDF，并且可以查看所有 Flow/Stream/Job



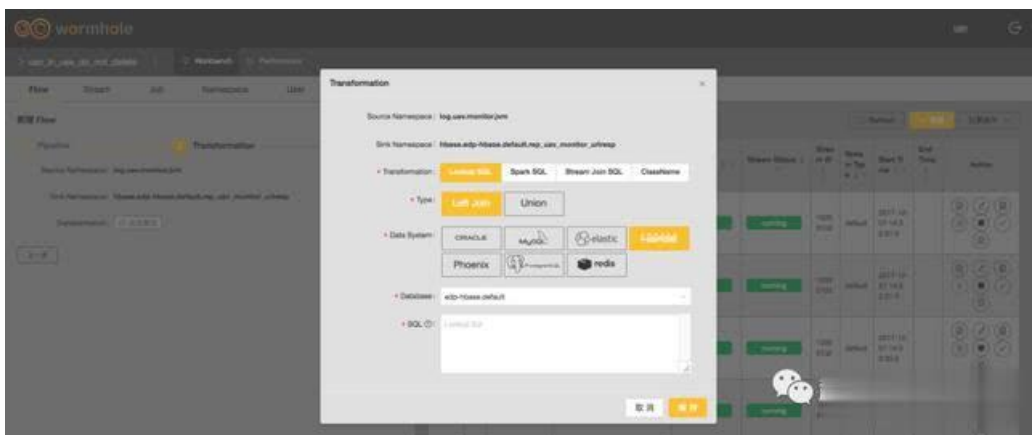
Admin 可以为 Project 分配 Namespace 资源/User 资源/UDF 资源/计算资源，以支持多租户资源隔离



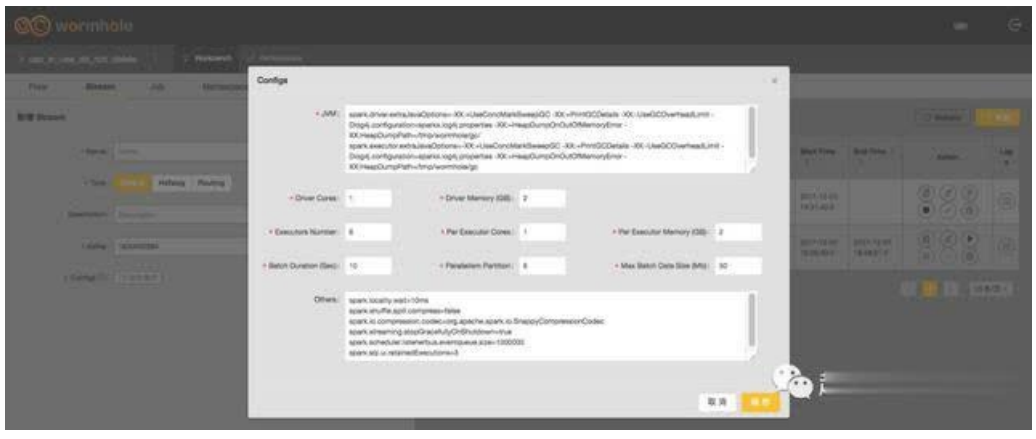
User 可以对自己有权限的 Project 进行开发实施和管理运维工作

User 可以通过简单配置步骤即可搭建起一个流式作业 pipeline（Flow），只需关注数据从哪来到哪去和如何转换处理

转换支持大部分流上作业常用场景，大部分工作可以通过配置 SQL 实现流上处理逻辑

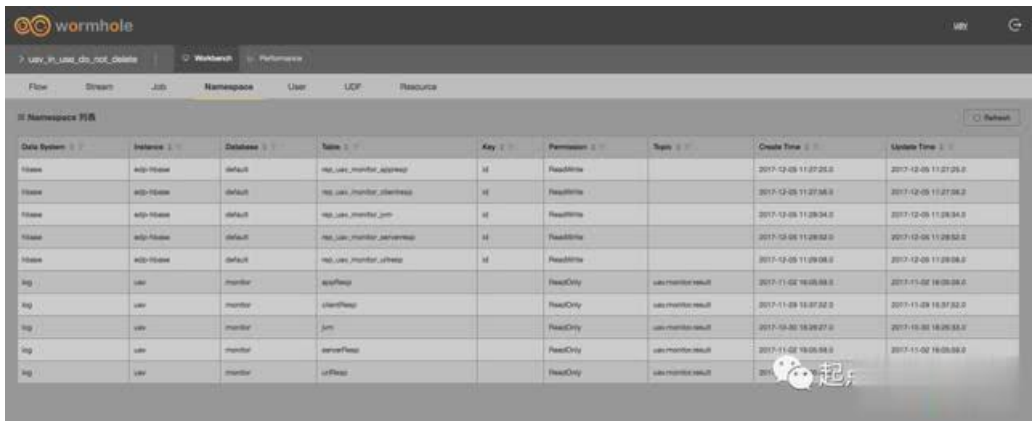


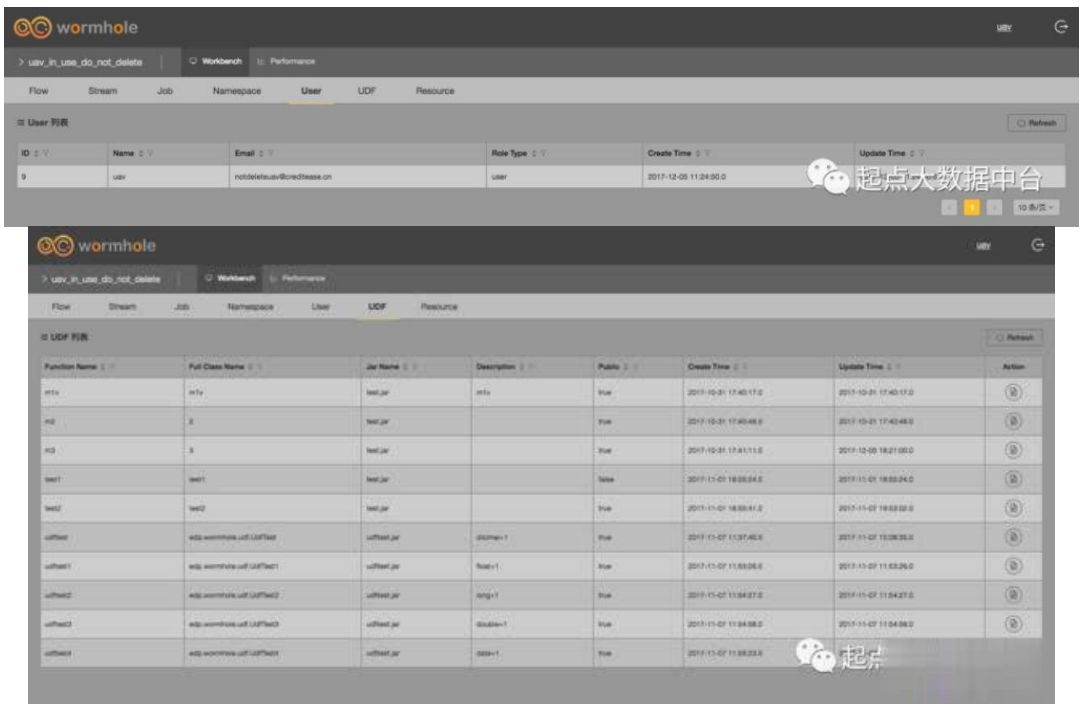
Wormhole 有 Flow 和 Stream 的概念，支持在一个物理 Stream（对应一个 Spark Stream）里通过并行处理多个逻辑 Flow，使得 User 可以更加精细灵活的利用计算资源，User 也可以对 Stream 进行精细化参数配置调整以更好平衡需求和资源



Wormhole 也支持批处理 Job，同样可以配置化实现处理逻辑并落到多个异构 Sink，Flow 和 Job 的配合可以很容易实现 Lambda 架构和 Kappa架构

User 可以查看 Project 相关的 Namespace/User/UDF/Resource





User还可以监控 Project 正在运行的所有 Flow/Stream 的吞吐和延迟

DBus + Wormhole如何协作?

把DBus做为数据抽取的管道，把所有的在业务系统的数据统一抽取到不同的Kafka Topic中；把Wormhole做为智能消费的路由系统，把Kafka不同Topic的数据做简单的处理之后根据上层数据开发的需求做不同的Sink：

针对离线数仓：Sink到Hive或者HDFS中

针对实时数仓：做初步转换后Sink到目标Kafka中，供上层实时引擎比如像Flink这样的引擎直接消费处理。

当然，DBus + Wormhole比我们想象的强大很多，单纯依靠这样的组合就能做到不写一行代码就能Cover 50%的数据开发场景。但是，对于我们需要构建一个通用、完善的DataWorks平台，只能选择每个系统的擅长的地方，择优用之。

把上层数据开发环节交给我们Part 1提到的Scriptis + Easy Scheduler组合，这样就更完美了。

全文总结

接受DBus + Wormhole的强大，也同时接受其强大背后的部署、运维的复杂性，我们拥有一个异常强大、易扩展、稳定的数据同步系统。而且是完全私有化，完全可控的，你值得拥有!

上一篇：联想、百度、蓝色光标投资智能撰稿机器人妙笔... 下一篇：UFS 3.0已成5G旗舰标配！一篇文章带你看看U...

相关阅读推荐：

2020视频平台剧集战"三重压力"

SEM要被DSP取代？扒一扒SEM、DSP与信息流广告的"...

CPU的主频真是越高越好吗?

大数据时代，新零售不能忘了"小数据"