**DTS设计文档**

## 修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| 修订版本号. | 修订内容简述 | 修订日期 | 修订人 |
| **1.0** | **概述，架构设计** | **2016.11.09** | **马进** |
|  |  |  |  |

## 评审记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 提交评审的修订版本号 | 评审意见/结果 | 评审形式 | 评审日期 | 评审人 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 第一章 概述

## 1. 应用场景

DTS全称Data Transfer System（暂定名），是网易大数据平台组计划研发的一套用于结构化数据传输和订阅的中间件系统，与DataStream一类的数据传输系统相比，它的特点是只作用于结构化数据，所以源端都是数据库，目标端可以是各种异构数据存储系统，未来DataStream用于结构化数据传输的DBCollector会慢慢退出历史舞台。

DTS可以且不限于适用以下几种场景：

1. 异构数据迁移

目前大多数应用对olap的解决方案是，用sqoop定期将线上的oltp业务数据库全量或按照某种过滤条件，导入到hive，hadoop或hbase中，这种做法一方面导入数据很慢，另一方面在导入数据的时段，一般每天凌晨，会对业务库产生巨大压力，并且造成网络拥塞。

而且，随着impala，kudu等更好的olap系统崛起，sqoop这类工具将不再实用，我们迫切需要一套对业务库压力小，实时性强，迁移过程平滑的结构化数据迁移解决方案。

1. 同构数据同步

异构数据迁移是将结构化数据从数据库中实时导入到olap数据仓库，目的是基于业务做大数据分析，查询，挖掘有价值的信息。还有一种实用数据迁移场景，用于异地灾备，多活，这种场景下，异地部署的业务一般不会变，因而底层数据库同构居多，我们将这种实时数据迁移的场景定义为“数据同步”，相比于数据迁移，数据同步对端都要支撑oltp业务，因而对实时性和一致性要求更高，再加上异地网络的不可控，需要压缩算法降低传输延迟，数据库如mysql自身提供的主备同步方案往往不是最优选择。

除了数据同步的实时性和一致性之外，对于一些复杂的异地多活案例，还需要解决星状拓扑和数据回环问题。

3．结构化数据订阅

数据库的目的是数据存储和持久化，它可以被动接收应用各种查询请求并予以回复，却不能主动告知外界自身状态的变化，数据订阅功能可以很好地弥补数据库的这个短板。结构化数据订阅**（以下简称数据订阅）**是指将数据库中每时每刻产生的数据增量变化实时拉取出来，主动交付给相关业务方，一般以SDK的方式提供服务。结构化数据订阅可以实现缓存异步更新，数据驱动的业务解耦等，例如在电商应用中，通过订阅订单数据，可以将下单流程中的库存，物流业务抽离出来，降低业务耦合度。

目前我们对DTS的需求主要在**异构数据迁移和数据订阅**上，数据迁移可以打通MySQL，oracle到各种olap系统的壁垒，同时也可以作为DDB 做数据重分布，在线扩缩容，在线修改表结构的底层依赖。数据订阅可以给云音乐，考拉这类大应用做业务治理：将数据驱动的业务线隔离出来，或者通过订阅数据库增量数据实现多机房缓存更新和淘汰。

## 2. 设计目标

目前DTS主要服务于网易内部产品，数据迁移功能作为BDMS，DDB的依赖组件，数据订阅功能开放给网易内部应用做缓存更新淘汰和异步业务解耦。

由于DTS会给多个系统做支撑，不可避免的会涉及到适配问题，我们希望DTS系统本身可以作为一个独立系统存在，有自己的运维工具，平台管理，节点管理等模块，通过RPC或http接口向BDMS或DDBAdmin提供服务。BDMS中集成DTS的管理界面，DDBAdmin将DDB的扩容缩容，在线修改表结构等操作转换为一个个单机数据库的迁移任务，并向DTS系统发起请求。

同时，DTS也要支持多租户，但是接入哪套租户认证还有待商榷。

DTS中包含的基本组件有：

* 管控模块/节点：节点管理，任务调度，元数据管理
* 执行模块/节点：迁移，订阅任务的具体执行模块
* Dashboard：可视化运维工具
* SDK：数据订阅组件

## 3. 功能需求

目前设计中，DTS支持数据迁移和数据订阅两类功能。源端先支持MySQL和Oracle，目的端先支持Kudu, MySQL, Oracle和DDB。

对外，DTS通过Dashboard以web的形式为产品管理员提供各项管理功能：

1. **租户认证**：由于DTS是面向多租户的，需要接入云计算或猛犸的账号系统，并将各个迁移和订阅任务按照租户统计和管理。
2. **任务调度**：管控节点负责迁移和订阅任务的创建，预检查和分发到执行节点。目前DTS要求迁移任务必须有一个源端实例和一个目的端实例，迁移和订阅对象以表为单位，一个任务可以包含一个源端实例内的多个表对象。
3. **任务监控**：迁移或订阅任务在执行过程中，会定时向管控节点上报任务执行状态，以及任务执行过程中产生的各个状态值，如MySQL增强迁移时源端的binlog位置点。DTS对任务的监控要多维度的：除了不同迁移过程，如全量和增量会产生不同的状态值之外，对每个迁移对象表也会分开记录状态值。
4. **任务管理：**用户可以通过管控平台将创建任务，或将迁移和订阅任务暂停，启动，停止或释放，停止后的任务再也无法启动，只会在系统中保留任务最后时刻的状态值，释放后的任务会彻底从DTS系统中删除。
5. **动态更改迁移对象**：无论是迁移任务还是订阅任务，都要求DTS具备动态更改迁移对象的功能，即可以在迁移或订阅任务执行过程中，增加或减少迁移对象表。

除了面向产品管理员的各种功能外，DTS还需要具备面向平台管理员的管理功能，主要是执行节点的管理和监控：

1. **Engine管理：**查看所有执行节点和相关状态信息，执行节点优雅下线，释放等。平台管理员功能，不按租户区分
2. **SDK管理：**查看订阅任务关联SDK节点列表，以及相关状态信息

## 4. 非功能需求

除了功能性需求外，NTS3.0对非功能需求的考量如下：

1. **管控高可用**：管控模块一般是中心组件，可采用冷备实现高可用
2. **任务高可用**：在执行节点宕机，或发生网络分区时，通过管控节点将任务迁移到其他执行节点重试，实现任务的自动take over

## 5. 竞品调研

网易DTS的竞品主有Oracle golden gate（以下简称OGG），阿里Otter，阿里DTS。

OGG是一个典型的数据迁移软件，是oracle为异构数据迁移场景提供的企业级解决方案，优点是支持的异构系统很多，缺点是不开源，没有平台化，部署和使用方法复杂。

Otter是阿里B2B开源的一套数据同步平台，对远距离，如中美机房的数据同步场景有较多的优化，只支持MySQL和Oracle。优点是有平台化，缺点是概念很多，使用方法较复杂，不支持全量迁移。

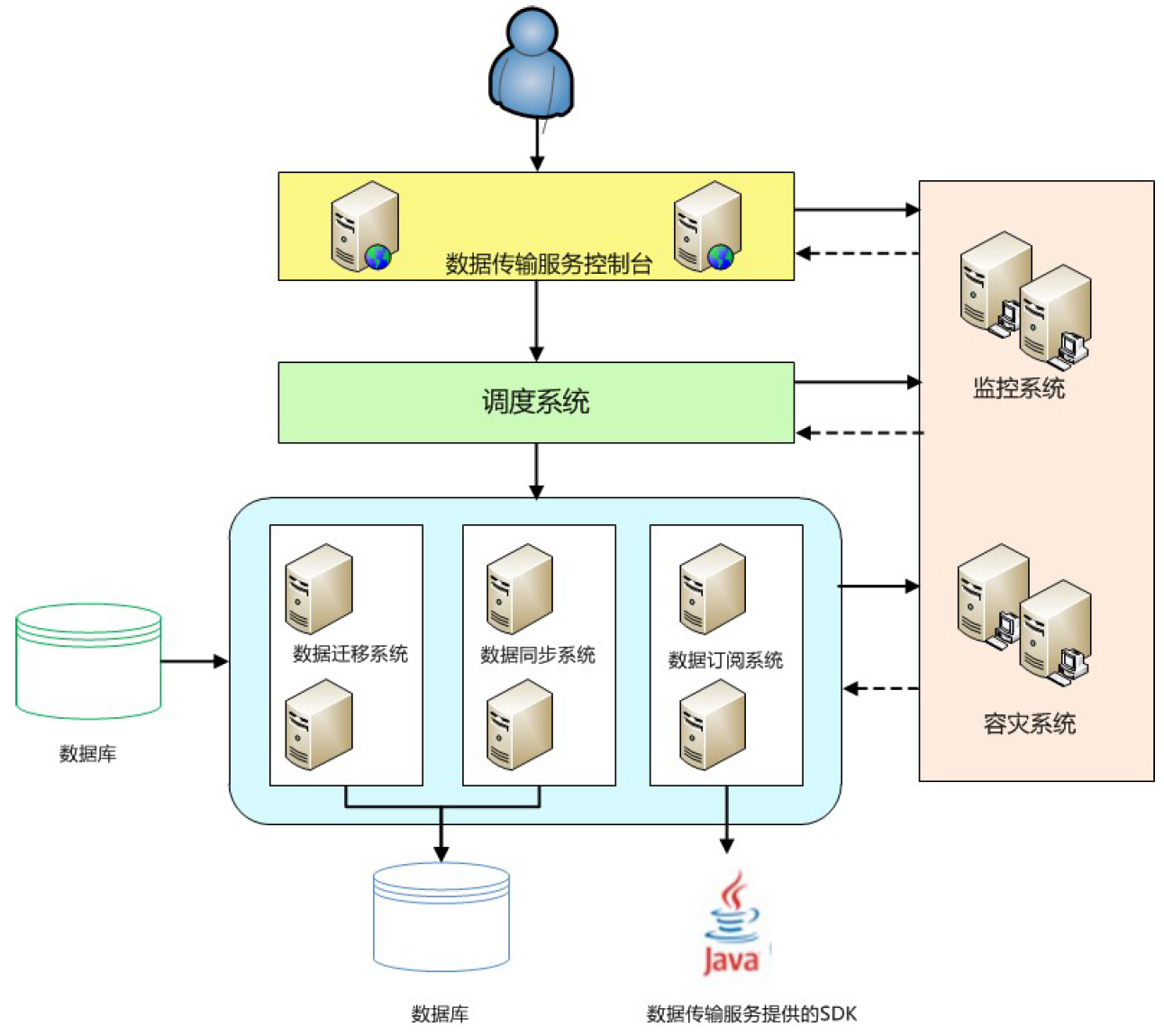
阿里DTS是阿里云在云端提供的一套数据迁移，数据同步和数据订阅的SaaS服务，必须承认阿里在数据库，数据传输领域已经走在了很靠前的位置。网易DTS在产品概念和形态上有很多借鉴于阿里DTS。阿里DTS应该是最早将数据迁移，数据同步和数据订阅三种结构化数据传输场景严格定义并提供相应云服务的供应商。



# 第二章 架构设计

网易DTS目前的设计目标首要是为网易内部产品提供数据迁移和订阅服务，但并不排除未来通过蜂巢提供公有云服务，在这方面，阿里DTS已经走在了行业前面。阿里DTS的产品定位与目标用户与我们的DTS基本一致，不同的是阿里DTS支持数据同步功能，并且支持的迁移和同步目标仅限于阿里的各类数据存储系统。而网易DTS虽然在设计上源端仅支持MySQL和Oracle，目标端支持的系统也没有阿里DTS多，但网易DTS旨在帮助应用解决实际的数据迁移和订阅问题，未来会根据应用实际需求支持更多的系统。

作为标杆性的竞品，我们先来看下阿里DTS的架构：



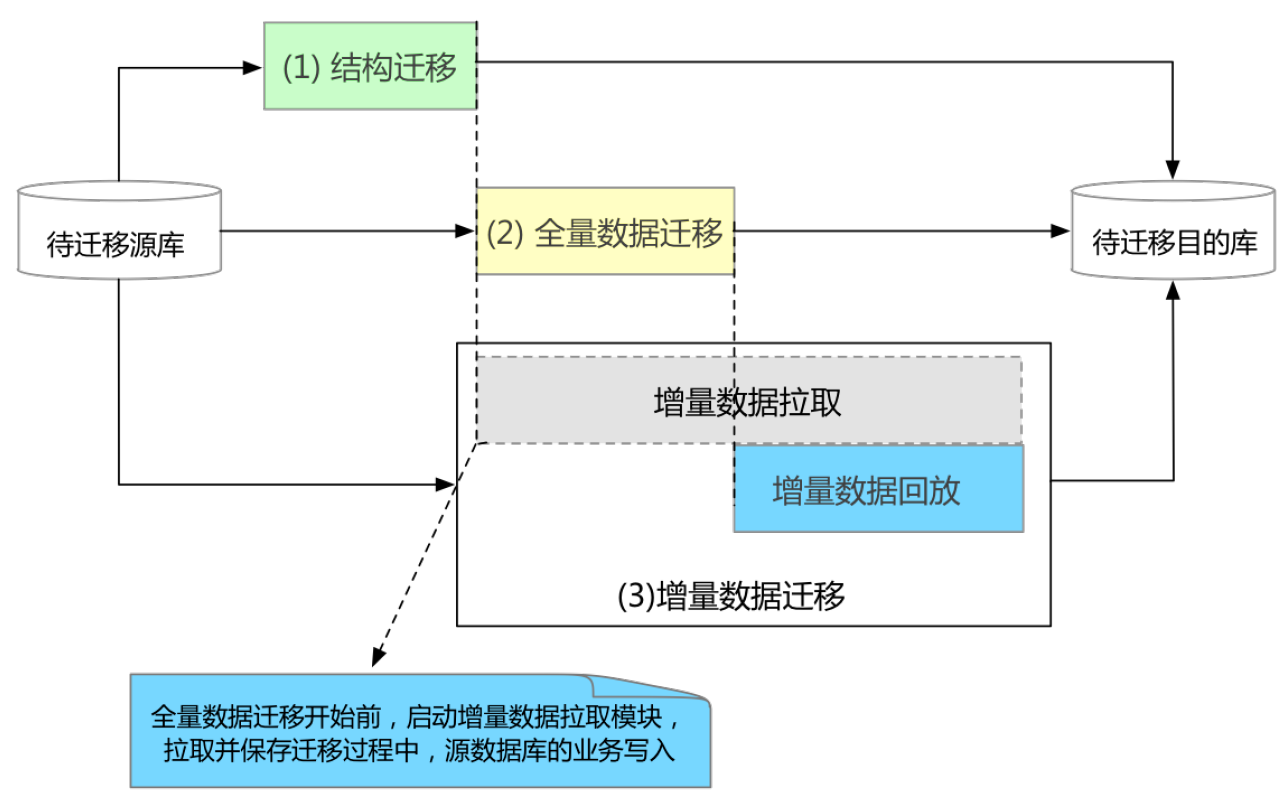
与真实的架构图相比，上图更像是一张概念架构图，即便如此，我们也可以看到阿里DTS和网易DTS在基础模块上的划分非常类似，我们有专门的Dashboard作为DTS的管理工具，与图种的“服务控制台”相对应，我们设计了center模块作为调度中心，同时也是整个系统的监控中心，作为中心模块，我们未来会用ZK一类的分布式协调组件实现高可用，这样center对应了“调度系统”和“监控系统”，ZK对应了“容灾系统”，图种的数据迁移子系统和数据订阅子系统分别对应了两种不同类型的engine。综上，上面的概念架构图完全适用于网易DTS，下面将分别介绍网易DTS中的各个模块和术语，数据迁移和数据订阅的基础架构以及全系统架构。

## 1. 术语与模块

|  |  |  |
| --- | --- | --- |
| 名词 | 解释 | 使用类型 |
| Engine | 执行节点统称，一个执行节点可执行多个任务，为了方便管理，同一个Engine不能同时执行迁移和订阅任务 | Both |
| SDK | SDK通常是指数据订阅中应用获取增量数据依赖的JAR包，在本文档中也可能指依赖SDK订阅数据的应用节点 | 数据订阅 |
| Center | DTS中的管控节点，接收所有Engine的状态上报，负责所有任务的管理和调度。 | Both |
| Dashboard | DTS的web运维工具，可以在Dashboard中创建任务，更改任务对象，查看任务各个对象，各个阶段的状态值等 | Both |
| 对象 | 本文档中的对象一般是指迁移和订阅对象，**DTS中以表为迁移和订阅单位**，这一方面是大部分应用场景所要求的，另一方面方便以表为维度做统计监控 | Both |
| Pipeline | 一个对象的迁移或订阅过程被称为Pipeline，Pipeline是任务状态统计的X维度，一个任务可以包含数十数百个Pipeline不等 | Both |
| Process | 一个任务虽然可以包含多个对象，但多个对象同时只会处于同一个过程，如所有迁移对象都处于全量复制过程，或增量复制过程，这些不同的过程抽象为Process的概念，Process是任务统计监控的Y维度 | Both |
| 结构迁移 | 指源端schema的迁移，即DDL的迁移。结构迁移同时作用于全量复制过程和增量复制过程 | 数据迁移 |
| 全量复制  /迁移 | 将源端的所有存量数据迁移到目标端的过程，全量复制默认串行执行，按照配置对象的顺序将源表逐个迁移到目标表，可以配置每批迁移的行数 | 数据迁移 |
| 增量复制  /迁移 | 将源端的增量数据实时迁移到目标端，或将源端的实时更改从元旦抽取出来，交付给订阅端。在迁移场景中，增量复制一般指迁移全量迁移过程中新产生的数据变更，在迁移不停止的情况下，增量迁移将源端的实时更改应用到目标端。而我们所说的数据订阅，专指订阅增量数据 | Both |

## 2. 数据迁移

数据迁移一般包括结构迁移，全量迁移和增量迁移三个过程（参考上节术语解释），以源端MySQL为例，三者的关系大致如下所示：



在一个完整的迁移过程中，执行节点会先将源端所有需要迁移的表结构在目标端中创建，即结构迁移，结构迁移完成后，会先启动增量数据的拉取模块，再开始全量迁移，这样做是为了保证全量迁移过程中的增量更新不会丢失。

全量迁移的原理是单个线程分批从源端拉取存量数据，通过目标系统的驱动将存量数据插入到目标端中，目前全量迁移提供的配置项比较简单，只有一次批量拉取数据的数量。而在全量迁移过程中可能出现两种情况：拉取对源端压力过高，或迁移过程过慢。对于前者，可以增加一个批量拉取的时间间隔配置项，减轻对源端的压力；对后者可以增加一个并发拉取的选项，以加速全量迁移速度。这两种配置都需要对业务情况做全面权衡，而单线程不间断全量迁移是一种普遍接受的做法。

与全量迁移相同，增量迁移也需要将增量数据从源端拉取到本地，再通过目标系统驱动应用到目标端中，不同的地方在于：增量数据拉取模块对不同的源端系统可能有截然不同的实现，例如对MySQL而言，增量拉取就是dump binlog，使用MySQL特有的通信协议，而对Oracle而言，可能是通过JDBC将CDC中的数据定时拉出来；其次，增量迁移中数据回放和数据拉取必须异步进行，因为源端的增量拉取往往比回放速度快很多，如果因回放速度慢而导致源端增量日志无法及时消费，可能影响源端的服务稳定性。另外，为了实现异步回放且不会OOM，拉取模块会把增量数据落盘，回放模块读盘来异步消费。DTS默认采用了Hamal的并行复制算法，用户在创建含有增量迁移的迁移任务时，需要指定增量迁移的并发度。

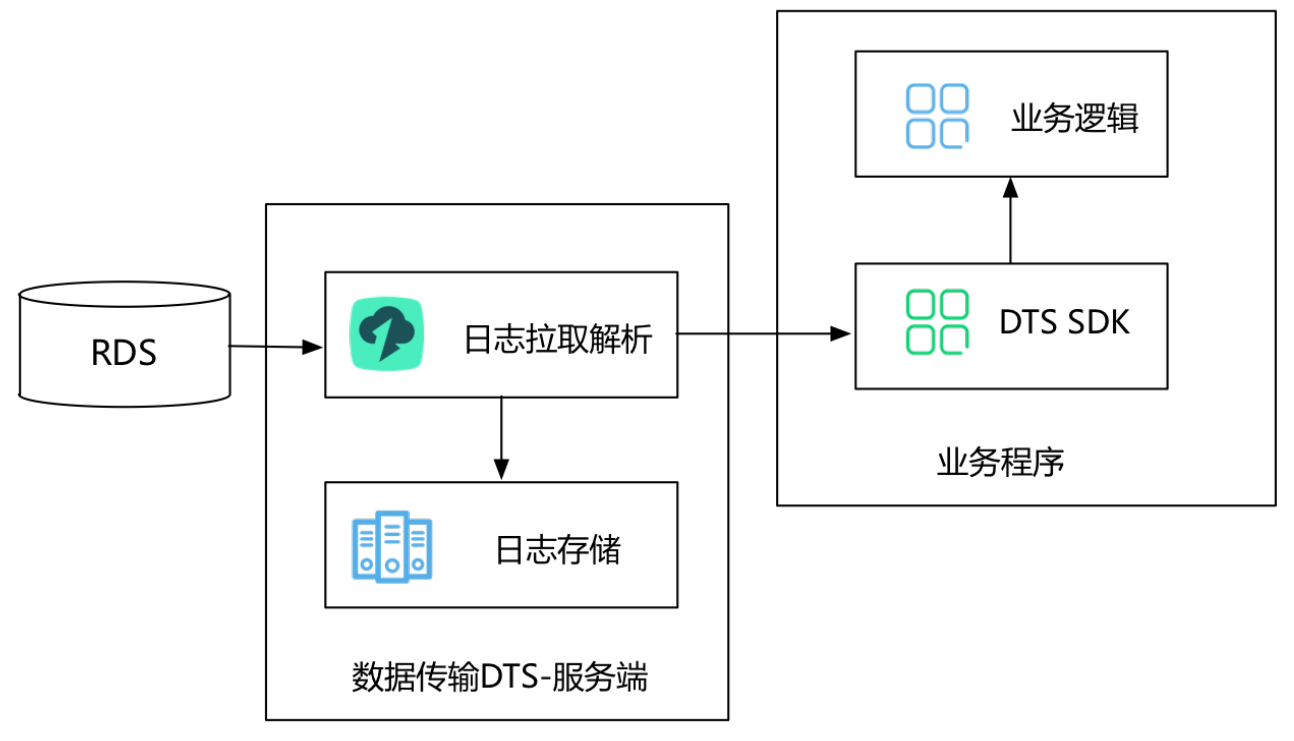
在上一节中有提到，结构迁移同时作用于全量复制和增量复制过程，所以结构迁移不能算作一个Process，而是作为全量迁移和增量迁移过程中需要handle的一个特性。全量迁移中，结构迁移是在真正的全量迁移开始之前完成，增量迁移中，engine要确保每个结构迁移必须发生在相关结构事件迁移之前。

考虑到结构迁移的业务复杂性：传统的DDL会锁表，而pt-osc一类的在线修改表结构机制也无法立即生效，这种结构迁移极容易造成整个迁移过程不可控。且结构迁移在一些没有schema的系统中是一个伪命题，例如HBase。所以我们建议前期先将结构迁移交给运维人员处理，这样一方面规避了结构迁移自动化给系统带来的不可控因素，另一方面也允许了schema不同但兼容的情况出现（当然也可以通过提供一个是否结构迁移的选项来实现）。

DTS一期在创建一个任务时可以选择是否全量迁移和增量迁移，系统允许对一个迁移任务只做全量迁移或增量迁移。

## 2. 数据订阅

数据订阅是将源端数据库的增量数据拉取出来，并通过DTS的SDK实时交付给应用端进行定制化处理，它的基础架构如下图所示：



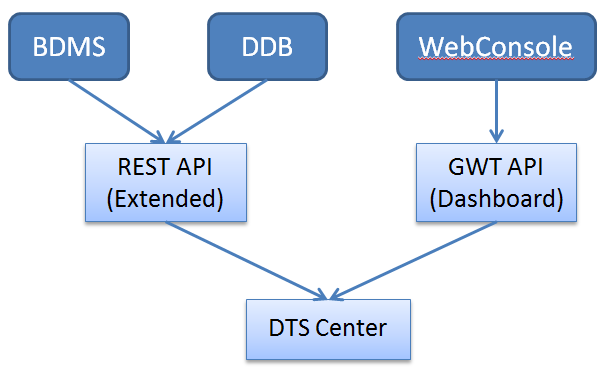
数据订阅与数据迁移的增量迁移过程类似，不过数据订阅的Apply过程是将增量日志转换为消息丢进消息队列，交由SDK处理。

由于数据库的增量更新必须保证有序，所以选用的消息队列必须可以保障消息的有序性，而下游消费数据的SDK同时只能有一个接收消息，否则可能出现一条数据先后的更新操作由两个SDK节点接收，顺序难以保障。

目前的方案选型是用kafka的单partition策略，日志拉取后是否保存在engine本地可选。

## 3. 可视化运维

DTS为用户提供WEB版Dashboard作为可视化管理工具，用户可以通过Dashbaord创建，监控和跟踪每个数据迁移和数据订阅任务。Dashboard是DTS独立于BDMS和DDB的工具，是DTS独立的入口，而DTS在作为一个独立服务的同时，也是大数据平台BDMS和DDB管理工具做数据迁移的依赖组件，为此我们需要为BDMS和DDB提供方便适配的REST接口。



如上图所示，我们在开发Dashboard时选用了vaadin作为WEB开发框架，vaadin内嵌了很多GWT的控件，与纯GWT相比，vaadin的强大之处在于是全JAVA开发，不用编辑html，类似于swing开发窗口程序，是前后端一站式开发的利器。而GWT与REST完全两种风格，这里我们把他们作为并列的组件。REST API为BDMS和DDB提供REST接口来适配（DDB由于我们自己维护，可以和Dashboard一样通过RPC与center交互），GWT API则以Dashboard的形式为DTS提供独立的管理服务。

在实现上，REST API和GWT API可以置于一个容器中。

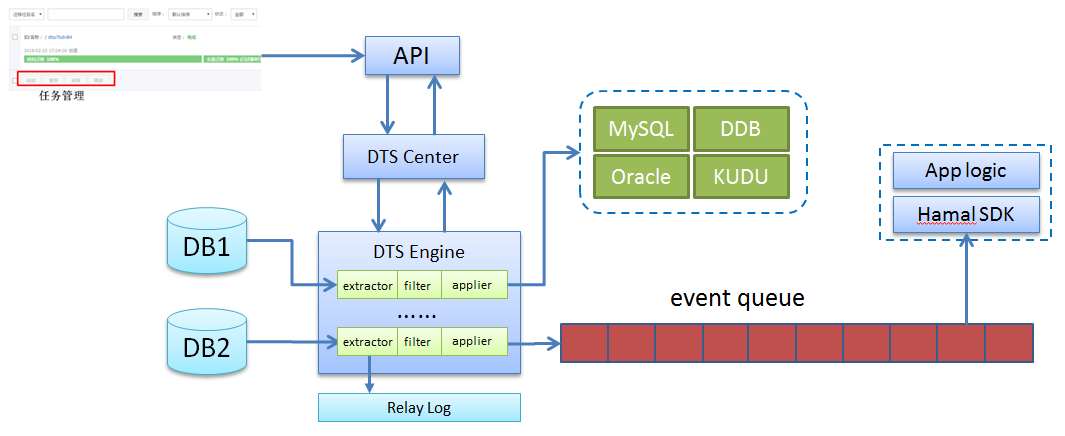
## 4. 平台接入

DTS未来可能会接入不同平台，不同平台有不同的用户认证服务，比如BDMS和蜂巢就是两种不同的用户体系。所谓平台接入就是接入不同的用户认证体系。

为了最大限度保持DTS的独立性，我们把DTS的用户认证放在API层，而不是在center中，center中的用户统一以int类型标志，并存储用户以及相关的平台信息，以供查阅。而实际的用户认证操作放在API层去扩展。这样平台接入只需要在API层进行适配，而不需要更改center和engine。

## 5. 总体架构

DTS的整体架构如下图所示：



API模块接收来自WEB或其他系统的请求，创建任务时，API节点收到请求后通过RPC的方式转发给center，center负责将任务描述信息记录到元数据库中。启动任务时，center先执行一些固有的预检查，如网络连通性检查，用户权限检查，然后选择合适的engine节点将下发。engine启动相应任务后，会定期将任务运行时状态上报给center，center再将这些运行时信息保存到元数据库中，以便center重启后可以从元数据库中恢复所有job信息。

Center作为DTS的中心模块，未来会使用ZK或keepAlived一类的高可用组件以冷备方式实现高可用。