**调研报告**

# 背景和目标

随着大数据时代的到来，数据的种类和来源越来越丰富和多样化。在维护数据完整性以及数据的传输和交互上，迫切的需要一款通用的数据同步工具，以便可以在多种数据源、多个数据接口之间自由同步和交换数据。

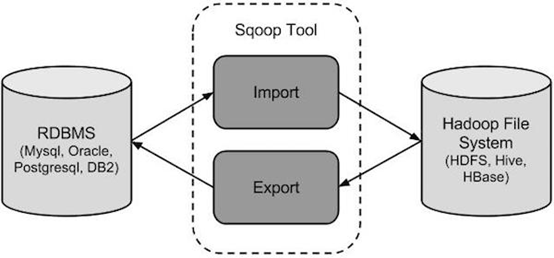
最初版本的数据同步工具功能较为单一，结构却相对复杂。只支持sqlserver数据库之间的数据同步，而且中间依赖sqlserver底层的bcp命令、ftp文件传输、zip文件解压缩等。由于程序环节较多，使用起来较为繁琐，且任何环节出错，都会导致整个同步任务失败；另外只支持单一关系型数据库sqlserver的功能略显单薄，且后续功能的增加难以在原有架构上进行扩展。

当前目标就是想使用一个通用的、可扩展的数据同步工具将当前已经存在的关系型数据库（MYSQL、SQLSERVER）以及其他多种数据源关联起来，并且可以任意同步各数据源上的数据。

# 成熟数据同步方案

## Apache开源软件：Sqoop

Sqoop(发音:skup)是一款开源的工具，主要用于在HADOOP(Hive)与传统的数据库(mysql、postgresql...)间进行数据的传递，可以将一个关系型数据库(例如 : MySQL ,Oracle ,Postgres等)中的数据导进到Hadoop的HDFS中，也可以将HDFS的数据导进到关系型数据库中。Sqoop专为大数据批量传输设计，能够分割数据集并创建Hadoop任务来处理每个区块。通过导入导出命令加配套参数控制操作。



Sqoop支持全量数据导入和增量数据导入（增量数据导入分两种，一是基于递增列的增量数据导入（Append方式）。二是基于时间列的增量数据导入（LastModified方式）），同时可以指定数据是否以并发形式导入。

## Kettle开源软件

Kettle是一款国外开源的ETL工具，纯java编写，可以在Window、Linux、Unix上运行，绿色无需安装，数据抽取高效稳定。存在两种脚本文件：transformation和job，transformation完成针对数据的基础转换，job则完成整个工作流的控制

Kettle 中文名称叫水壶，该项目的程序员MATT 希望把各种数据放到一个壶里，然后以一种指定的格式流出。

Kettle的Spoon有丰富的Steps可以组装开发出满足多种复杂应用场景的数据集成作业，方便实现全量、增量数据同步。缺点是通过定时运行，实时性相对较差。

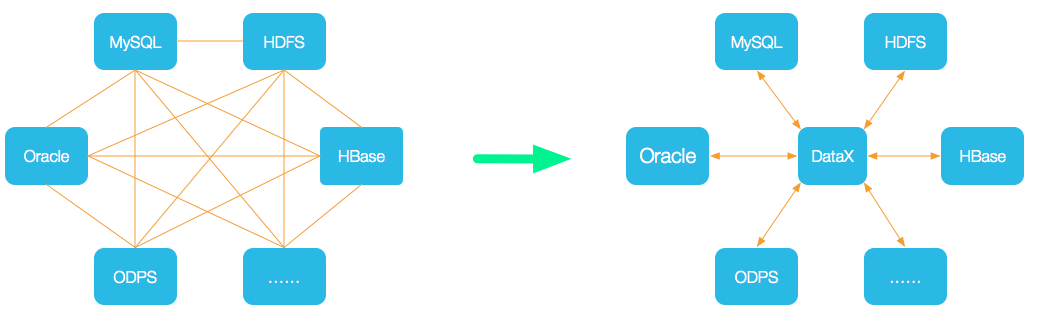
Kettle家族目前包括4个产品：Spoon、Pan、CHEF、Kitchen。

* SPOON:允许你通过图形界面来设计ETL转换过程（Transformation）；
* PAN:允许你批量运行由Spoon设计的ETL转换 (例如使用一个时间调度器)。Pan是一个后台执行的程序，没有图形界面；
* CHEF:允许你创建任务（Job）。 任务通过允许每个转换，任务，脚本等等，更有利于自动化更新数据仓库的复杂工作。任务通过允许每个转换，任务，脚本等等。任务将会被检查，看看是否正确地运行了；
* KITCHEN:允许你批量使用由Chef设计的任务 (例如使用一个时间调度器)。KITCHEN也是一个后台运行的程序。

# DataX

## DataX简介

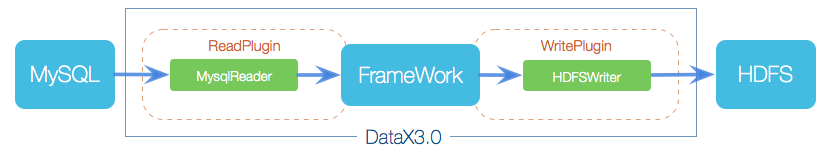
​ DataX 是一个异构数据源离线同步工具，致力于实现包括关系型数据库(MySQL、Oracle等)、HDFS、Hive、ODPS、HBase、FTP等各种异构数据源之间稳定高效的数据同步功能。



当加入新的数据源的时候，只要指定合适的插件，都可以实现与现有系统和数据源之间的无缝衔接与数据同步。如果不存在合适的插件，DataX也支持自建插件，扩展性优异。

* **性能**：DataX核心使用读写线程分离加上数据通道的设计方案；数据同步性能优异。
* **通用：**DataX目前支持7种关系型数据、4种阿里云存储、5种NoSQL数据库、4种无结构数据（这里ElasticSearch算作无结构数据，而不是NoSQL）之间的同步；通用性较好。
* **扩展：**DataX采用插件式开发架构设计；支持开发者继承Reader、Writer接口实现对其他数据源的读写操作，可扩展性优异。
* **健壮：**DataX采用了多种Job与Task校验机制、可以控制读取写入速度、可以在线程和进程级别分别对任务进行重试；容错性较强，整体较为健壮。
* **使用成本：**使用DataX只需要配置Job即可，指定数据源和目标数据地址以及读写插件即可；使用成本相对较低。

## DataX框架设计



DataX本身作为离线数据同步框架，采用Framework + plugin架构构建。将数据源读取和写入抽象成为Reader/Writer插件，纳入到整个同步框架中。

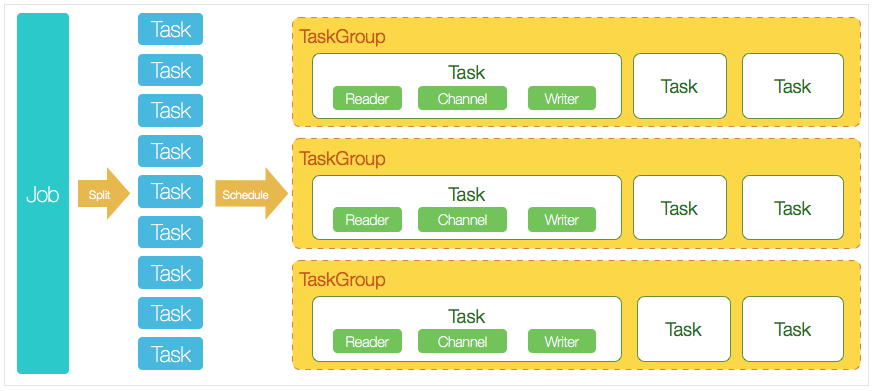
* **Reader**：Reader为数据采集模块，负责采集数据源的数据，将数据发送给Framework。
* **Writer**： Writer为数据写入模块，负责不断向Framework取数据，并将数据写入到目的端。
* **Framework**：Framework用于连接reader和writer，作为两者的数据传输通道，并处理缓冲，流控，并发，数据转换等核心技术问题。

## DataX已支持的插件体系

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类型 | 数据源 | Reader(读) | Writer(写) | 文档 |
| RDBMS 关系型数据库 | MySQL | √ | √ | [读](https://github.com/alibaba/DataX/blob/master/mysqlreader/doc/mysqlreader.md) 、[写](https://github.com/alibaba/DataX/blob/master/mysqlwriter/doc/mysqlwriter.md) |
|  | Oracle | √ | √ | [读](https://github.com/alibaba/DataX/blob/master/oraclereader/doc/oraclereader.md) 、[写](https://github.com/alibaba/DataX/blob/master/oraclewriter/doc/oraclewriter.md) |
|  | SQLServer | √ | √ | [读](https://github.com/alibaba/DataX/blob/master/sqlserverreader/doc/sqlserverreader.md) 、[写](https://github.com/alibaba/DataX/blob/master/sqlserverwriter/doc/sqlserverwriter.md) |
|  | PostgreSQL | √ | √ | [读](https://github.com/alibaba/DataX/blob/master/postgresqlreader/doc/postgresqlreader.md) 、[写](https://github.com/alibaba/DataX/blob/master/postgresqlwriter/doc/postgresqlwriter.md) |
|  | DRDS | √ | √ | [读](https://github.com/alibaba/DataX/blob/master/drdsreader/doc/drdsreader.md) 、[写](https://github.com/alibaba/DataX/blob/master/drdswriter/doc/drdswriter.md) |
|  | 达梦 | √ | √ | [读](https://github.com/alibaba/DataX/blob/master) 、[写](https://github.com/alibaba/DataX/blob/master) |
|  | 通用RDBMS(支持所有关系型数据库) | √ | √ | [读](https://github.com/alibaba/DataX/blob/master) 、[写](https://github.com/alibaba/DataX/blob/master) |
| 阿里云数仓数据存储 | ODPS | √ | √ | [读](https://github.com/alibaba/DataX/blob/master/odpsreader/doc/odpsreader.md) 、[写](https://github.com/alibaba/DataX/blob/master/odpsswriter/doc/odpswriter.md) |
|  | ADS |  | √ | [写](https://github.com/alibaba/DataX/blob/master/adswriter/doc/adswriter.md) |
|  | OSS | √ | √ | [读](https://github.com/alibaba/DataX/blob/master/ossreader/doc/ossreader.md) 、[写](https://github.com/alibaba/DataX/blob/master/osswriter/doc/osswriter.md) |
|  | OCS | √ | √ | [读](https://github.com/alibaba/DataX/blob/master/ocsreader/doc/ocsreader.md) 、[写](https://github.com/alibaba/DataX/blob/master/ocswriter/doc/ocswriter.md) |
| NoSQL数据存储 | OTS | √ | √ | [读](https://github.com/alibaba/DataX/blob/master/otsreader/doc/otsreader.md) 、[写](https://github.com/alibaba/DataX/blob/master/otswriter/doc/otswriter.md) |
|  | Hbase0.94 | √ | √ | [读](https://github.com/alibaba/DataX/blob/master/hbase094xreader/doc/hbase094xreader.md) 、[写](https://github.com/alibaba/DataX/blob/master/hbase094xwriter/doc/hbase094xwriter.md) |
|  | Hbase1.1 | √ | √ | [读](https://github.com/alibaba/DataX/blob/master/hbase11xreader/doc/hbase11xreader.md) 、[写](https://github.com/alibaba/DataX/blob/master/hbase11xwriter/doc/hbase11xwriter.md) |
|  | MongoDB | √ | √ | [读](https://github.com/alibaba/DataX/blob/master/mongoreader/doc/mongoreader.md) 、[写](https://github.com/alibaba/DataX/blob/master/mongowriter/doc/mongowriter.md) |
|  | Hive | √ | √ | [读](https://github.com/alibaba/DataX/blob/master/hdfsreader/doc/hdfsreader.md) 、[写](https://github.com/alibaba/DataX/blob/master/hdfswriter/doc/hdfswriter.md) |
| 无结构化数据存储 | TxtFile | √ | √ | [读](https://github.com/alibaba/DataX/blob/master/txtfilereader/doc/txtfilereader.md) 、[写](https://github.com/alibaba/DataX/blob/master/txtfilewriter/doc/txtfilewriter.md) |
|  | FTP | √ | √ | [读](https://github.com/alibaba/DataX/blob/master/ftpreader/doc/ftpreader.md) 、[写](https://github.com/alibaba/DataX/blob/master/ftpwriter/doc/ftpwriter.md) |
|  | HDFS | √ | √ | [读](https://github.com/alibaba/DataX/blob/master/hdfsreader/doc/hdfsreader.md) 、[写](https://github.com/alibaba/DataX/blob/master/hdfswriter/doc/hdfswriter.md) |
|  | Elasticsearch |  | √ | [写](https://github.com/alibaba/DataX/blob/master/elasticsearchwriter/doc/elasticsearchwriter.md) |

## DataX核心架构设计

DataX 3.0 **开源版本**支持单机多线程模式完成同步作业运行（目前开源版本不支持分布式多进程运行），下面根据DataX作业生命周期的时序图，从整体架构设计非常简要说明DataX各个模块相互关系。



#### 核心模块介绍

1. DataX完成单个数据同步的作业，我们称之为Job，DataX接受到一个Job之后，将启动一个进程来完成整个作业同步过程。DataX Job模块是单个作业的中枢管理节点，承担了数据清理、子任务切分(将单一作业计算转化为多个子Task)、TaskGroup管理等功能。
2. DataXJob启动后，会根据不同的源端切分策略（例如分表或者主键切分），将Job切分成多个小的Task(子任务)，以便于并发执行。Task便是DataX作业的最小单元，每一个Task都会负责一部分数据的同步工作。
3. 切分多个Task之后，DataX Job会调用Scheduler模块，根据配置的并发数据量，将拆分成的Task重新组合，组装成TaskGroup(任务组)。每一个TaskGroup负责以一定的并发运行完毕分配好的所有Task，默认单个任务组的并发数量为5。
4. 每一个Task都由TaskGroup负责启动，Task启动后，会固定启动Reader—>Channel—>Writer的线程来完成任务同步工作。
5. DataX作业运行起来之后， Job监控并等待多个TaskGroup模块任务完成，等待所有TaskGroup任务完成后Job成功退出。否则，异常退出，进程退出值非0

#### DataX调度流程

举例来说，用户提交了一个DataX作业，并且配置了20个并发，目的是将一个100张分表的mysql数据同步到odps里面。 DataX的调度决策思路是：

1. DataXJob根据分库分表切分成了100个Task。
2. 根据20个并发，DataX计算共需要分配4个TaskGroup（因为默认单个TaskGroup的并发度是5）。
3. 4个TaskGroup平分切分好的100个Task，每一个TaskGroup负责以5个并发共计运行25个Task。

## DataX本地调试与开发

### 下载开源版本代码

DataX开源版本的源代码地址：

<https://github.com/alibaba/DataX>

可以直接下载zip包，也可以使用git clone命令checkout：git clone git@github.com:alibaba/DataX.git

### IDEA中调试源码

使用idea打开项目所在目录，构建为maven项目，DataX的实际项目名称为datax-all。

|  |  |  |
| --- | --- | --- |
| Module类型 | 实际意义 | Module名称 |
| 通用 | 包含通用功能、类、工具等 | Common |
| 核心 | 处理同步任务的核心模块 | Core |
| 插件 | 每个Task内部负责具体的读写操作 | XXXReader、XXXWriter |

需要事先准备好一个core.json以及job.json；这两个文件分别是任务的核心配置以及任务详情配置。两个配置文件的大致格式如下所示：

Core.json

1. {
2. "entry": {
3. "jvm": "-Xms1G -Xmx1G",
4. "environment": {}
5. },
6. "common": {
7. "column": {
8. "datetimeFormat": "yyyy-MM-dd HH:mm:ss",
9. "timeFormat": "HH:mm:ss",
10. "dateFormat": "yyyy-MM-dd",
11. "extraFormats":["yyyyMMdd"],
12. "timeZone": "GMT+8",
13. "encoding": "utf-8"
14. }
15. },
16. "core": {
17. "dataXServer": {
18. "address": "http://localhost:7001/api",
19. "timeout": 10000,
20. "reportDataxLog": **false**,
21. "reportPerfLog": **false**
22. },
23. "transport": {
24. "channel": {
25. "class": "com.alibaba.datax.core.transport.channel.memory.MemoryChannel",
26. "speed": {
27. "byte": -1,
28. "record": -1
29. },
30. "flowControlInterval": 20,
31. "capacity": 512,
32. "byteCapacity": 67108864
33. },
34. "exchanger": {
35. "class": "com.alibaba.datax.core.plugin.BufferedRecordExchanger",
36. "bufferSize": 32
37. }
38. },
39. "container": {
40. "job": {
41. "reportInterval": 10000
42. },
43. "taskGroup": {
44. "channel": 5
45. },
46. "trace": {
47. "enable": "false"
48. }
50. },
51. "statistics": {
52. "collector": {
53. "plugin": {
54. "taskClass": "com.alibaba.datax.core.statistics.plugin.task.StdoutPluginCollector",
55. "maxDirtyNumber": 10
56. }
57. }
58. }
59. }
60. }

Job.json

1. {
2. "job": {
3. "content": [
4. {
5. "reader": {
6. "name": "sqlserverreader",
7. "parameter": {
8. "column": [
9. "\*"
10. ],
11. "connection": [
12. {
13. "jdbcUrl": [
14. "jdbc:sqlserver://192.168.1.51\\sql01;DataBaseName=MBD2\_CITY\_MAIN"
15. ],
16. "table": [
17. "tb\_cfg\_broad\_dp\_mm\_200424"
18. ]
19. }
20. ],
21. "password": "dtauser",
22. "username": "dtauser",
23. "where": ""
24. }
25. },
26. "writer": {
27. "name": "mysqlwriter",
28. "parameter": {
29. "column": [
30. "\*"
31. ],
32. "connection": [
33. {
34. "jdbcUrl": "jdbc:mysql://192.168.2.101:3306/rgx\_test?useUnicode=true&characterEncoding=utf8",
35. "table": [
36. "tb\_cfg\_broad\_dp\_mm\_200427"
37. ]
38. }
39. ],
40. "password": "dtauser",
41. "preSql": [
42. ""
43. ],
44. "username": "dtauser",
45. "writeMode": "insert"
46. }
47. }
48. }
49. ],
50. "setting": {
51. "errorLimit": {
52. "percentage": 0.02,
53. "record": 0
54. },
55. "speed": {
56. "byte": 10485760
57. }
58. }
59. }
60. }

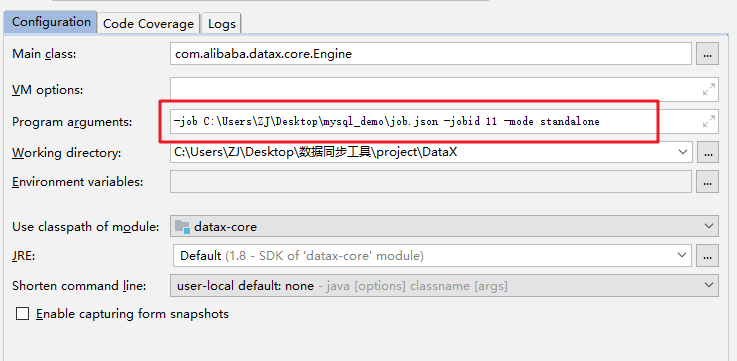
1、如果core.json不在项目目录，可以在CoreConstant.java中修改地址：



2、DataX插件式架构中，运行中会去加载reader和writer对应的plugin的jar包，然后加载实现类。但是调试过程中我们想调试插件代码，那么可以把插件的依赖加入到core模块中，即在core module的pom.xml文件中加入相关依赖。



3、启动项目的入口在com.alibaba.datax.core. Engine;启动前在configuration中Program arguments中配置相关启动参数：



参数详情为：-job C:\Users\ZJ\Desktop\mysql\_demo\job.json -jobid 11 -mode standalone

以上步骤完成后，就可以在IDEA中调试DataX代码了。

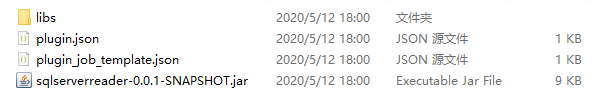
### MAVEN构建后脚本运行

1、通过maven打包：mvn -U clean package assembly:assembly -Dmaven.test.skip=true

2、打包完成后在项目的target目录，会有一个datax目录，这个就是datax软件的运行目录



* bin：DataX的运行脚本所在目录，都是python脚本，所以需要保证程序所在机器有python环境
* conf：DataX的配置文件所在目录，例如上文讲到的core.json文件就在此目录
* job：可以用来存放job.json等文件
* plugin：包含reader和writer两个目录；分别存放插件的读和写的相关插件，例如sqlserverReader内部的目录结构：



其中plugin.json主要记录了插件class文件的全限定名，以便于core module加载插件的时候可以按图索骥。

## 示例：RDBMS数据源的同步

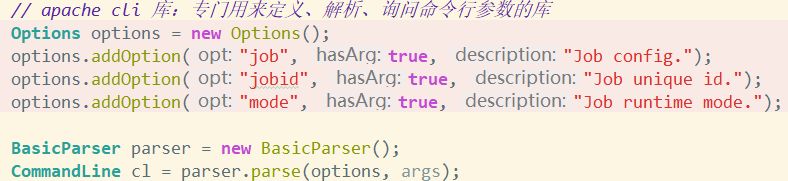
### 逻辑流程分析

下面从传统关系型数据库的同步上来讲解一下DataX的数据同步实现细节

程序的入口在Core模块的com.alibaba.datax.core. Engine。Main函数核心代码只有一行代码：Engine.entry(args); 进入此函数…

这个函数主要完成四件事情：

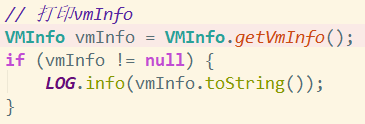
* 读取程序启动参数



* 初始化程序配置，并对插件地址等信息做了特殊处理，方便后续的使用



* 打印虚拟机状态



* 启动此次任务



Engine的start方法主要完成了配置的绑定，并且根据配置决定启动JobContainer或者是TaskGroupContainer，在当前示例中，container为JobContainer

进入container的start方法

这个方法基本概括了一个同步任务的执行全过程（此处暂不分析DayRun为真的情况，此功能开源版本应该未完善）；下面分方法一一讲解同步任务的各个执行步骤：

* preHandle

如果在配置中指定了job.preHandler.pluginType，那么继续从配置中拿到job.preHandler.pluginName；然后以LoadUtil.loadJobPlugin方法根据pluginType和pluginName加载配置名称为plugin.${pluginType}.${pluginName}的相关配置，根据配置中的path属性找到jar包位置，进而生成一个具体的plugin实例，执行此实例的preHandle方法。

* init

读取reader和writer的相关配置，然后与上文一样的加载插件的方式将二者实例化为plugin并执行它们的init方法。在RDBMS的reader或者writer中init方法主要负责对批量读写的fetchSize和batchSize等参数进行设置

* prepare

执行reader和writer的prepare方法，这里需要注意的是：在DataX中，每一个plugin的内部都有两个子plugin，一个是AbstractJobPlugin的实现，一个是AbstractTaskPlugin的实现；两个抽象plugin都继承于AbstractPlugin。具体在什么阶段加载的什么plugin需要看在加载中使用了什么方法，LoadUtil的loadJobPlugin方法负责加载jobPlugin，而LoadUtil的loadTaskPlugin则负责加载taskPlugin。所以在此处其实执行的jobPlugin的prepare方法。

另外需要介绍的是在RDBMS的数据同步中，所有具体的操作都是交给两个工具类实现的，它们是com.alibaba.datax.plugin.rdbms.reader. CommonRdbmsReader、com.alibaba.datax.plugin.rdbms.writer. CommonRdbmsWriter。它们都是模块plugin-rdbms-util的工具类。

* split

顾名思义，此方法主要负责切分任务。在RDBMS数据同步中，DataX目前可以按照读写配置的table数量，读表的主键切分任务。在本示例中，只能切分一个任务。具体任务的切分实现可以查看代码实现细节

* schedule

重头戏来了，这个方法就开始实际同步数据了。主要使用AbstractScheduler的schedule方法，然后继续使用ProcessInnerScheduler的startAllTaskGroup方法。因为在前一步分割了任务，所以这里的Job其实已经已经被分为了TaskGroup了。具体的Task的执行交给了TaskExecutor，TaskExecutor开启了两个子线程，一个负责读，一个负责写。

* post

Job的数据好同步工作完成后，执行指定的饿后置操作

* postHandle

同PreHandle

* invokeHooks

## 开发新插件

请参考DataX的插件开发指南

<https://github.com/alibaba/DataX/blob/master/dataxPluginDev.md>