

Datax增量任务调度程序说明

一、背景

Datax是一个批量数据抽取框架/工具，提供了常用的Reader、Writer等组件的实现，可以实现在不同数据源之间进行批量数据迁移。由于Datax本身并没有提供作业调度的功能，因此开发了VsDatax-Scheduler程序。

二、总体设计

VsDatax Scheduler包含两个模块：增量变量生成和定时任务调度

1、增量变量生成

增量变量生成调用“基于增量变量的数据增量抽取调度框架”实现（详见参考《基于增量变量的数据增量抽取调度框架设计》）。

在配置上除了“增量变量”相关的配置部分，还包括Datax Job的元数据相关的配置项。完整的配置示例如下：

1) Datax Job元数据配置项

```
#####
#The following items relate to the configurations of job iteself.
#作业 id: 区分作业的唯一性标识
job.meta.base.jobid=job_demo_onlytime
# 作业组名.用于quartz任务调度
job.meta.base.jobgroupname=job.group1
#作业的定时任务的cron表达式
job.meta.base.cronexpression=*/6 * * * * ?
#Datax程序的根路径
job.meta.base.datax.home=D:/work/java_res/datax/datax_aio
#包含作业定义的datax job脚本文件
job.meta.base.job.script=D:/work/projs/incubator/vsetl/vsdatax/scheduler/vsdatax-
scheduler/scripts/oracle2stream_onlytime.json
#Job的调用方式，有三种类型：线程方式、进程方式、ssh远程调用方式
job.meta.base.invoke.way=process
#启动作业的 python命令名。根据作业的宿主机的操作系统进行配置，例如python、python2、python3等
job.meta.base.invoke.python.cmd=python2
#Datax提供的运行job的启动脚本。通常是$DATA_HOME/bin/datax.py
job.meta.base.invoke.datax.script=D:/work/java_res/datax/datax_aio/bin/datax.py
```

2) 运行状态配置项

```
#唯一标识Job状态存储信息的主键。
# 例如：对于采用文件存储 Job Status的实现，key一般采用状态文件的绝对路径名；
# 对于采用数据库存储Job Status的实现，key可以是用于唯一标识一条记录的主键，这里主键需要配置自动保持
# 全局唯一性
job.status.persist.key=d:/tmp/job_demo_onlytime.job
```

3) 增量变量配置项

```
#####
以下配置从本地配置项中获取起始值，从状态数据文件中获取开始值。
结束值从系统全局变量中获取（见datax的job配置文件），不需要在这里额外配置。如下：
"querySql": [
  "select * from kh_ydkh where jk_jkclsj>to_date('${begin}','yyyy-mm-dd hh24:mi:ss') and
  jk_jkclsj<=to_date('${_SYS_NOW}','yyyy-mm-dd hh24:mi:ss') and jk_dabs_id<10"
]
#####

#自定义的可变配置项。key名需要以job.var开头
#从本地配置项中提取开始位置的变量解析器实现
job.var.impl.begin=vsincr.variable.impl.LocalBeginVarResolver
```

```
#初始值。第一次抽取时的起始位置
# 初始值属性的配置优先级顺序:init.val,init.query.
job.var.conf.begin.init.val=1970-01-01 01:01:01
#状态记录里的上一次结束值的变量名字。默认是__SYS_NOW
job.var.conf.begin.lastend.var.name=__SYS_NOW
```

2、定时任务调度

定时任务调度基于Quartz实现了常规的任务调度，并提供基于Rest Service接口的作业管理功能，包括：对job的增删改查、调用指定的Job等。

三、使用说明

1、配置说明

1) beanDef/scheduler.bean.properties

配置系统使用的实现类

```
#作业管理器实现类。一般不改动
jobManager=vsdatax.scheduler.schedule.DataxJobManager
#作业配置管理器实现类。可以根据需要改动。系统提供了基于文件的配置管理程序和基于数据库的配置管理实现。
jobConfMgr=vsdatax.scheduler.jobconf.FileBasedJobConfMgr
#作业的生命周期管理服务类。默认是不支持事务的实现类
jobLifecycleService=vsdatax.scheduler.service.NonTransJobLifecycleService
#作业调用服务类。一般不改动
jobExecuteService=vsdatax.scheduler.service.JobExecuteService
#作业状态数据管理类。可以根据需要改动。系统提供了基于文件的和基于数据库的状态数据管理实现
#jobStatusMgr=vsincr.jobstatus.FileJobStatusMgr
jobStatusMgr=vsdatax.scheduler.jobstatus.JdbcJobStatusMgr
```

2) jobs

如果采用基于文件的配置管理程序 (如1中所述),则所有的job配置文件需要存放在该目录下。系统启动时将遍历该目录下所有的job配置, 并做相应处理。

3)appCfg.properties

系统的全局相关配置

```
#是否启用 jdbc连接池。当使用基于数据库的状态管理或者job配置管理时, 需要启用
jdbc数据库连接池
scheduler.jdbc.enable=true
# 基于文件的增量jobs配置文件所在的根路径。可选配置, 默认值是
${vsapp.workdir}/jobs
incr.job.conf.dir=${vsapp.workdir}/jobs
#####
#Datax Job元数据的默认值配置
meta.conf.default.datax.home=D:/work/java_res/datax/datax_aio
meta.conf.default.env.encoding=utf-8
meta.conf.default.invoke.way=process
meta.conf.default.python.cmd=python2
```

4) log4j.properties

log4j相关配置

5) quartz.properties

quartz相关配置

6) scheduler_jdbc.properties

当使用基于数据库的状态管理或者job配置管理时,启动的连接池配置

```
driverClassName=com.mysql.jdbc.Driver
url=jdbc:mysql://mydev:3306/datax_scheduler
username=xxx
password=xxxxxx
filters=stat
initialSize=2
maxActive=300
maxWait=60000
timeBetweenEvictionRunsMillis=60000
minEvictableIdleTimeMillis=300000
validationQuery=SELECT 1
testWhileIdle=true
```

```
testOnBorrow=false
testOnReturn=false
poolPreparedStatements=false
maxPoolPreparedStatementPerConnectionSize=200
```

7)sysvar.register.properties

系统全局变量配置。在该文件里配置的全局变量将被自动传递给datax job使用。

```
#在Job任务中引用系统变量需要加上前缀__SYS__
#UUID全局变量，生成UUID值
UUID=vscommons.vutils.sysvar.UUIDVarEvaluator
#系统当前时间全局变量，生成系统当前时间
NOW=vsincr.sysvar.StrNowVarEvaluator
```

8) vsdatax-server.properties

datax rest 服务相关配置。

```
rest.host=0.0.0.0
rest.port=9080
server.name=vsdatax
```

2、调用说明

1) 需要预先配置环境变量vsapp.workdir，告诉程序配置文件所在的根目录，如下示例：

-Dvsapp.workdir=D:\work\projs\incubator\vsetl\vsdatax\scheduler\vsdatax-scheduler\dataxConf

2) 启动程序入口vsdatax.scheduler.erest.startup.ERestStartup

启动程序做了以下事情：

- 遍历所有的增量Job配置，与quartz里管理的job按照最后更新时间进行比对，如果是新job,则创建或更新Job；如果是旧的job，则跳过。
- 启动HTTP REST 服务器，对外发布针对JOB管理调度的服务。

四、配置项示例

本节提供了几个典型的增量抽取场景的示例。为方便展示，采用基于文件的配置形式。

1、采用系统时间戳作为增量变量的结束值，配置上使用默认值以尽量减少不必要的配置项。

```
#####
#采用系统时间戳作为增量变量的结束值。对应的datax的job的query sql如下所示：
# "querySql": [
# "select * from users where last_modified_time>to_date('${begin}','yyyy-mm-dd hh24:mi:ss')
and last_modified_time<=to_date('${__SYS_NOW}','yyyy-mm-dd hh24:mi:ss') "
# ]
#
#配置上使用默认值以尽量减少不必要的配置项
#####
#The following items relate to the configurations of job itself.
#作业 id: 区分作业的唯一性标识
job.meta.base.jobid=job_onlytime_brief
# 作业组名.用于quartz任务调度
job.meta.base.jobgroupname=job.group1

#作业的定时任务的cron表达式
job.meta.base.cronexpression=*/6 * * * * ?

#包含作业定义的datax job脚本文件
job.meta.base.job.script=D:/work/projs/incubator/vsetl/vsdatax/scheduler/vsdatax-
scheduler/scripts/oracle2stream_onlytime.json

#唯一标识Job状态存储信息的主键。
# 例如：对于采用文件存储 Job Status的实现，key一般采用状态文件的绝对路径名；
# 对于采用数据库存储Job Status的实现，key可以是用于唯一标识一条记录的主键。这里主键值必须具备
全局唯一性。
job.status.persist.key=d:/tmp/job_onlytime_brief.job
#####
```

```

#自定义的可变配置项。key名需要以job.var开头
#从本地配置项中提取开始位置的变量解析器实现
job.var.impl.begin=vsincr.variable.impl.LocalBeginVarResolver
#####
#数据库解析程序用到的自定义配置项
#初始值。第一次抽取时的起始位置
# 初始值属性的配置优先级顺序:init.val,init.query.
job.var.conf.begin.init.val=1970-01-01 01:01:01

```

2、从数据库里获取增量变量的初始值和结束值

```

#####
#从数据库里获取增量变量的初始值和结束值。对应的dax的job的query sql如下所示：
# "querySql": [
# "select * from users where last_modified_time>to_date('${begin}','yyyy-mm-dd hh24:mi:ss')
and last_modified_time<=to_date('${end}','yyyy-mm-dd hh24:mi:ss') "
# ]
#####
#The following items relate to the configurations of job iteself.
#作业 id: 区分作业的唯一性标识
job.meta.base.jobid=job_begin_end_both_in_db
#dax系统运行环境的编码，默认utf-8
job.meta.base.env.encoding=utf-8
# 作业组名.用于quartz任务调度
job.meta.base.jobgroupname=job.group1

#作业的定时任务的cron表达式
job.meta.base.cronexpression=*/6 * * * * ?
#Dax程序的根路径
job.meta.base.dax.home=D:/work/java_res/dax/dax_aio
#包含作业定义的dax job脚本文件
job.meta.base.job.script=D:/work/projs/incubator/vset1/vsdax/scheduler/vsdax-
scheduler/scripts/oracle2stream.json
#Job的调用方式，有三种类型：进程方式(process)、线程方式(thread)、SSH远程调用模式(ssh)。默认是
process
job.meta.base.invoke.way=process
#启动作业的 python命令名。根据作业的宿主机的操作系统进行配置，例如python、python2、python3等
job.meta.base.invoke.python.cmd=python2
#Dax提供的运行job的启动脚本。通常是$DATA_HOME/bin/dax.py
job.meta.base.invoke.dax.script=D:/work/java_res/dax/dax_aio/bin/dax.py

#唯一标识Job状态存储信息的主键。
# 例如：对于采用文件存储 Job Status的实现，key一般采用状态文件的绝对路径名；
# 对于采用数据库存储Job Status的实现，key可以是用于唯一标识一条记录的主键。这里主键值必须具
备全局唯一性

job.status.persist.key=d:/tmp/job_begin_end_both_in_db.status

#####
#自定义的可变配置项。key名需要以job.var开头
#用于RDBMS的提取开始位置的变量解析器实现
job.var.impl.begin=vsincr.scheduler.rdbms.var.DbBeginVarResolver

#用于RDBMS的提取结束位置的变量解析器实现
job.var.impl.end=vsincr.scheduler.rdbms.var.DbEndVarResolver
#RDBMS的数据库连接用户名
job.var.conf.db.conn.user=xxxxxx
#RDBMS的数据库连接密码
job.var.conf.db.conn.password=xxxxxx
#RDBMS的数据库连接url
job.var.conf.db.conn.url=jdbc:oracle:thin:@//xxxxxx:1521/dbsrv2
#RDBMS的数据库连接的jdbc驱动
job.var.conf.db.conn.driver=oracle.jdbc.driver.OracleDriver
#####
#####
#数据库解析程序用到的自定义配置项
#初始值。第一次抽取时的起始位置

```

```
# 初始值属性的配置优先级顺序:init.val,init.query.
#job.var.conf.begin.init.val=
#获取初始值的查询语句
#The type of the sql's return value must be string
job.var.conf.begin.init.query=select to_char( (min(last_modified_time)-
numtodsinterval(1,'second')),'yyyymmdd hh24:mi:ss') from users
#状态记录里的上一次结束值的变量名字。默认是__SYS_NOW。
job.var.conf.begin.lastend.var.name=end
#获取结束值的查询语句
#The type of the return value must be string
job.var.conf.end.query=select to_char(max(last_modified_time),'yyyymmdd hh24:mi:ss') from
users
```

3、SSH远程调用模式

需要在上述的配置项中增加SSH相关配置项，SSH相关配置项如下示例：

```
#ssh相关配置#####
#目标主机名
job.meta.base.ssh.host=master
#目标主机端口
job.meta.base.ssh.port=22
#目标主机SSH登录名
job.meta.base.ssh.userName=root
#目标主机SSH登录密码
job.meta.base.ssh.password=xxxxxx
#目标主机的操作系统
job.meta.base.ssh.os=linux
#目标主机SSH连接超时时间设置
job.meta.base.ssh.timeout=1000
```

完整示例如下：

```
#####
#从数据库里获取增量变量的初始值和结束值。对应的datax的job的query sql如下所示：
# "querySql": [
# "select * from users where last_modified_time>to_date('${begin}','yyyymmdd hh24:mi:ss')
and last_modified_time<=to_date('${end}','yyyymmdd hh24:mi:ss') "
# ]
#####
#The following items relate to the configurations of job iteself.
#作业 id: 区分作业的唯一性标识
job.meta.base.jobid=job_begin_end_both_in_db
#datax系统运行环境的编码，默认utf-8
job.meta.base.env.encoding=utf-8
# 作业组名.用于quartz任务调度
job.meta.base.jobgroupname=job.group1

#作业的定时任务的cron表达式
job.meta.base.cronexpression=*/6 * * * * ?
#Datax程序的根路径
job.meta.base.datax.home=D:/work/java_res/datax/datax_aio
#包含作业定义的数据x job脚本文件
job.meta.base.job.script=D:/work/projs/incubator/vsetl/vsdatax/scheduler/vsdatax-
scheduler/scripts/oracle2stream.json
#Job的调用方式，有三种类型：进程方式(process)、线程方式(thread)、SSH远程调用模式(ssh)。默认是
process
job.meta.base.invoke.way=ssh
#####
#ssh相关配置#####
job.meta.base.ssh.host=master
job.meta.base.ssh.port=22
job.meta.base.ssh.userName=root
job.meta.base.ssh.password=xxxxxx
job.meta.base.ssh.os=linux
job.meta.base.ssh.timeout=1000
#####

#启动作业的 python命令名。根据作业的宿主机的操作系统进行配置，例如python、python2、python3等
job.meta.base.invoke.python.cmd=python2
#Datax提供的运行job的启动脚本。通常是$DATA_HOME/bin/datax.py
```

```
job.meta.base.invoke.datax.script=D:/work/java_res/datax/datax_aio/bin/datax.py
```

#唯一标识Job状态存储信息的主键。

例如：对于采用文件存储 Job Status的实现，key一般采用状态文件的绝对路径名；

对于采用数据库存储Job Status的实现，key可以是用于唯一标识一条记录的主键。这里主键值必须具备全局唯一性

```
job.status.persist.key=d:/tmp/job_begin_end_both_in_db.status
```

```
#####
```

#自定义的可变配置项。key名需要以job.var开头

#用于RDBMS的提取开始位置的变量解析器实现

```
job.var.impl.begin=vsincr.scheduler.rdbms.var.DbBeginVarResolver
```

#用于RDBMS的提取结束位置的变量解析器实现

```
job.var.impl.end=vsincr.scheduler.rdbms.var.DbEndVarResolver
```

#RDBMS的数据库连接用户名

```
job.var.conf.db.conn.user=xxxxxx
```

#RDBMS的数据库连接密码

```
job.var.conf.db.conn.password=xxxxxx
```

#RDBMS的数据库连接url

```
job.var.conf.db.conn.url=jdbc:oracle:thin:@//xxxxxx:1521/dbsrv2
```

#RDBMS的数据库连接的jdbc驱动

```
job.var.conf.db.conn.driver=oracle.jdbc.driver.OracleDriver
```

```
#####
```

```
#####
```

#数据库解析程序用到的自定义配置项

#初始值。第一次抽取时的起始位置

初始值属性的配置优先级顺序:init.val,init.query.

```
#job.var.conf.begin.init.val=
```

#获取初始值的查询语句

#The type of the sql's return value must be string

```
job.var.conf.begin.init.query=select to_char( (min(last_modified_time)-  
numtodsinterval(1,'second')),'yyyymmdd hh24:mi:ss') from users
```

#状态记录里的上一次结束值的变量名字。默认是__SYS_NOW。

```
job.var.conf.begin.lastend.var.name=end
```

#获取结束值的查询语句

#The type of the return value must be string

```
job.var.conf.end.query=select to_char(max(last_modified_time),'yyyymmdd hh24:mi:ss') from  
users
```

五、使用步骤示例

1、用例说明

从t_book表中抽取数据。id是增量字段，将id作为增量抽取的增量变量。

表结构定义如下：

```
CREATE TABLE `t_book` (  
  `id` bigint(20) NOT NULL,  
  `book_name` varchar(32) NOT NULL,  
  `book_desc` text,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

数据示意如下：

id	book_name	book_desc
1	天渊	弗洛。文奇著名科幻小说
2	银河帝国	阿西莫夫著名科幻小说
3	机器人系列	阿西莫夫著名科幻小说之一
4	机器人系列-裸阳	裸阳2
5	天空的孩子	深渊系列

2、使用示例

需要预先配置环境变量vsapp.workdir，告诉程序配置文件所在的根目录，如下示例：

-Dvsapp.workdir=D:\work\projs\incubator\vsetl\vsdatax\scheduler\vsdatax-scheduler\dataxConf
以下用\${vsapp.workdir}环境变量值指向配置文件的根目录。

1)配置增量作业文件的存储位置，假定存储在\${vsapp.workdir}/jobs_pub_demo 目录

修改配置文件\${vsapp.workdir}/appCfg.properties，如下：

基于文件的增量jobs配置文件所在的根路径。可选配置，默认值是\${vsapp.workdir}/jobs

incr.job.conf.dir=\${vsapp.workdir}/jobs_pub_demo

2)增加datax job脚本。假定文件路径名为：

D:/work/projs/incubator/vsetl/vsdatax/scheduler/vsdatax-scheduler/scripts_pub_demo/books_num_incr_demo.json

配置内容如下：

```
{
  "job": {
    "setting": {
      "speed": {
        "channel": 5
      }
    },
    "content": [
      {
        "reader": {
          "name": "mysqlreader",

          "parameter": {
            "username": "root",
            "password": "xxxxxx",
            "column": ["*"],
            "where": " id >${begin} and id<=${end}",
            "connection": [
              {
                "table": [
                  "t_book"
                ],
                "jdbcUrl": [
                  "jdbc:mysql://mydev:3306/hyt_demo?useUnicode=true"
                ]
              }
            ]
          }
        },
        "writer": {
          "name": "streamwriter",
          "parameter": {
            "encoding": "UTF-8",
            "print": true
          }
        }
      }
    ]
  }
}
```

3)在\${vsapp.workdir}/jobs_pub_demo下添加增量作业文件job_book_num_demo.job

配置文件如下：

```
#####
#The following items relate to the configurations of job iteself.
#作业 id: 区分作业的唯一性标识
job.meta.base.jobid=job_book_num_demo
# 作业组名.用于quartz任务调度
job.meta.base.jobgroupname=job.group1

#作业的定时任务的cron表达式
job.meta.base.cronexpression=*/6 * * * * ?
#Datax程序的根路径
job.meta.base.datax.home=D:/work/java_res/datax/datax_aio
```

```

#包含作业定义的datax_job脚本文件
job.meta.base.job.script=D:/work/projs/incubator/vsetl/vsdatax/scheduler/vsdatax-
scheduler/scripts_pub_demo/books_num_incr_demo.json
#invoke way includes three options:process,thread,ssh
job.meta.base.invoke.way=process
#启动作业的 python命令名。根据作业的宿主机的操作系统进行配置，例如python、python2、python3等
job.meta.base.invoke.python.cmd=python2
#Datax提供的运行job的启动脚本。通常是$DATA_HOME/bin/datax.py
job.meta.base.invoke.datax.script=D:/work/java_res/datax/datax_aio/bin/datax.py

#唯一标识Job状态存储信息的主键。
job.status.persist.key=d:/tmp/job_book_num_demo.status
#####
以下配置实现功能：
第一次运行时从配置项中获取配置的job.var.conf.begin.init.val值作为开始值；
此后从运行状态数据中获取上次的结束值作为本次运行的开始值；
从数据库中查询max值作为本次运行的结束值。
#####

#从本地配置项中提取开始位置的变量解析器实现
job.var.impl.begin=vsincr.variable.impl.LocalBeginVarResolver
#状态文件中对应的结束变量值的Key的名字，默认__SYS_NOW
job.var.conf.begin.lastend.var.name=end
#用于RDBMS的提取结束位置的变量解析器实现
job.var.impl.end=vsincr.scheduler.rdbms.var.DbEndVarResolver
#RDBMS的数据库连接用户名
job.var.conf.db.conn.user=root
#RDBMS的数据库连接密码
job.var.conf.db.conn.password=xxxxx
#RDBMS的数据库连接url
job.var.conf.db.conn.url=jdbc:mysql://mydev:3306/hvt_demo?useUnicode=true
#RDBMS的数据库连接的jdbc驱动
job.var.conf.db.conn.driver=com.mysql.jdbc.Driver
#####

#####
#数据库解析程序用到的自定义配置项
#初始值。第一次抽取时的起始位置
# 初始值属性的配置优先级顺序:init.val,init.query.
job.var.conf.begin.init.val=0
#获取结束值的查询语句
#The type of the return value must be string
job.var.conf.end.query=select max(id ) from t_book

```

4)运行单脚本测试

系统提供了测试程序用于执行单个增量job。调用vsdatax.scheduler.job.JobLocalInvokerTest，传入增量job配置文件名。

注意：需要配置java环境变量，如下示例：

```
-Dvsapp.workdir=D:\work\projs\incubator\vsetl\vsdatax\scheduler\vsdatax-scheduler\dataxConf
```

测试类如下：

```

public class JobLocalInvokerTest {
    public static void main(String[] args) {
        String jobConfFile = "D:\\work\\projs\\incubator\\vsetl\\vsdatax\\scheduler\\vsdatax-
scheduler\\dataxConf\\jobs_pub_demo\\job_book_num_demo.job";

        EnvInitiator.initEnv();
        JobInvokerByFileFacade.invoke(jobConfFile, null, 0);
    }
}

```

运行结果如下：

```

2019-07-31 14:57:14.243 [0-0-0-reader] INFO CommonRdbmsReader$Task - Finished read record
by Sql: [select * from t_book where (id >0 and id <=5)
]jdbcUrl:
1 天渊 弗洛。文奇著名科幻小说
2 银河帝国 阿西莫夫著名科幻小说
3 机器人系列 阿西莫夫著名科幻小说之一
4 机器人系列-裸阳 裸阳2
5 天空的孩子 深渊系列
2019-07-31 14:57:24.206 [job-0] INFO JobContainer -

```


任务启动时刻	:	2019-07-31 14:57:13
任务结束时刻	:	2019-07-31 14:57:24
任务总计耗时	:	10s
任务平均流量	:	6B/s
记录写入速度	:	0rec/s
读出记录总数	:	5
读写失败总数	:	0

执行成功后会在所配置的状态文件job_book_num_demo.status中记录如下执行结果，用于下次的增量执行。

```
{"__SYS_NOW":"2019-07-31  
14:57:11","__SYS_UUID":"815759de465248da86c864f4fb4b9b65","end":"5","begin":"0"}
```