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.vsutils.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('${begin}','yyyy-mm-dd hh24:mi:ss')
and last_modified_time<=to_date('${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
#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datax 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=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。
# 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字
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')),'yyyy-mm-dd 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),'yyyy-mm-dd 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('${begin}','yyyy-mm-dd hh24:mi:ss')
and last_modified_time<=to_date('${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
#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datax 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字
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')),'yyyy-mm-dd 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),'yyyy-mm-dd hh24:mi:ss') from
users
```