SAS® GLOBAL FORUM 2016

IMAGINE. CREATE. INNOVATE.

Advanced ETL Scheduling Techniques

with SAS® Data Integration Studio and IBM Platform LSF

Angus Looney
Senior Solution Architect

Capgemini UK.

Email: Angus.Looney@capgemini.com
Twitter: @AngusLooney

#SASGF

Advanced ETL Scheduling Techniques

with SAS® Data Integration Studio and IBM Platform LSF

Angus Looney, Senior Solution Architect, Capgemini UK

ABSTRACT

- **Platform LSF** is the scheduling and orchestration tool that is bundled with Data Integration Studio and SAS Grid.
- A range of basic scheduling and triggering approaches are supported by LSF.
- Using **SAS Data Integration Studio** you can achieve much more sophisticated scheduling and triggering approaches based on incoming unprocessed data and currently running and historic ETL flows.

INTRODUCTION

ETL processes are built using **SAS Data Integration Studio** by creating the jobs to carry out the necessary Extract, Transform and Load (ETL) steps, then using **SAS Management Console** to assemble jobs into flows to orchestrate the jobs in the correct logical sequence. Flows are then deployed to **LSF**, where **Process Manager** is used to schedule their execution. When flows are executed, LSF takes care of running the jobs on the available SAS server(s).

LSF Process Manager has a basic set of conditions it can use to schedule when flows should be started, or "triggered".

•Time-based - day(s) of week/month, specific hours/minutes

•File-based -existence or arrival of specified filename masks

•Manual - human operator triggers flow

These triggering conditions have some draw backs:

•Time-based - Ideal for fixed time triggering, but it's not flexible. What if data turns up late, or early?

•File-based - responsive, but generates a lot of file system polling and logging, and doesn't work well where new files turn up frequently.

•Manual - allows ultimate flexibility in deciding when to trigger or not, but would require a full time operator.

However, this e-poster will show how it's easy to build a rules-based, automated flow triggering system using **SAS Data Integration Studio**.

Building “FlowMaster”

In order to build our rules-based flow triggering system, we need a set of key capabilities, to be able to:

1. Determine details of unprocessed raw data for each ingest flow
2. Build a “rules engine” to decide when a flow needs to be triggered (based on unprocessed data)
3. Interact with LSF to:
   - Determine what flows are already running
   - Determine if named flows are deployed and are not on hold
   - Trigger named flow

Let’s look at each of these capabilities in turn.

1. Determine status of raw data files to be ingested.

A variety of approaches can be used, based on the prevailing approach to managing and tracking raw data.

Examples include simple file and directory based schemes and holding data on raw data in control tables.

What’s essential is that we can provide data on availability of raw data to input into the rules engine.

The richer the information that can be provided, the more powerful the rules that can be constructed.

Ideally, you require the data on the files that are in each of these three states:

- **Unprocessed** – newly arrived data awaiting ingest
- **In Process** – data currently being ingested
- **Processed,** - data that has been successfully ingested

grouped by Ingest subject or data source.

The information that needs to be gathered includes

- timestamp of oldest file
- timestamp of youngest file
- total count of files
- total size of files

The system will be recording the date and time whenever it triggers flows, and this information will be made available to the rules engine as well.

This is the key data the rules engine at the heart of the system is going to process in order make decisions on what flows to trigger.

Advanced ETL Scheduling Techniques

with SAS® Data Integration Studio and IBM Platform LSF

Angus Looney, Senior Solution Architect, Capgemini UK

Building “FlowMaster” continued

2. Rules Engine

We can build “rules engine” in SAS Data Integration Studio using a **Data Validation** node.

First you combine the data for the prevailing flow triggering rules into a data table with a single row per rule, combined with the status of relevant raw data files to be ingested.

The Data Validation node calculates a value for a “decision to trigger” field using a Boolean expression that evaluates to true or false, based on whatever rule is specified.

The Boolean expression combines values in selected fields in the input data depending on the specific rule for each row, to return a decision on whether to trigger the relevant flow.

For example – a rule to trigger a flow if there was unprocessed data for a source and there wasn’t any data “in process” i.e. current being processed – would look something like:

```
        (Unprocessed_Count > 0 and In_Process_Count = 0)
```

where “Unprocessed_Count” was the number of files awaiting processing, and “In_Process_Count” was the number of files being processed.

So, using this rule, if there are some files awaiting processing, and none being processed, this expression evaluates to True, and relevant flow is tagged as ready to be triggered.

A range of example rules are discussed on the next slide.

3. Interacting with LSF

LSF provides the **Flow Manager** application as a user interface to monitor and manage the execution of flows, allowing users to:

- examine deployed flows
- determine which flows are currently running
- trigger flows manually

However, as all of the functionality of Flow Manager is available using command line utilities, we can call these using SAS code.

For our Data Integration Studio based system, we can use User Transforms to encapsulate these command line calls and turn their results into data that we can use with our rules engine.

(See the list of caveats on following slide).

Building “FlowMaster” continued

Determine what flows are running

Command: `jflows –u user_name –s state`

e.g.

```
  jflows –u all – s Running
```

lists all the flows that are running (for all users), with the following output:

```
ID      USER              NAME                    STATE
146     CORP\batchuser    *\batchuser:ABCD_Full Running
147     CORP\batchuser    *\batchuser:QRST_Full Running
148     CORP\batchuser    *\batchuser:WXYZ_Full Running
```

Determine status of a flow

Command: `jdefs –u username flowname`

e.g.

```
 jdefs –u batchuser ABCD_Full
```

returns details of flow (if it exists) and its Status as “Release” or “OnHold”, with the following output:

```
NAME         USER              STATUS       FLOW_IDS
ABCD_Full    CORP\batchuser    Release      599(Done)
                                            614(Done)
                                            625(Done)
                                            636(Done)
```

Trigger a Flow

Command: `jtrigger –u user_name flow_name`

e.g.

```
 jtrigger –u batchuser ABCD_Full
```

will start the flow “ABCD_Full” for user “batchuser”, with the following output:

```
Flow <CORP\batchuser:ABCD_Full> is triggered: Flow id <715>.
```

Advanced ETL Scheduling Techniques

with SAS® Data Integration Studio and IBM Platform LSF

Angus Looney, Senior Solution Architect, Capgemini UK

Example triggering rules

Triggering Rules: examples

Basic Rule: “**process new data**”

Trigger flow if there is new raw data, and no “in process” data
and it’s not already running

More Advanced Rule: “**let the dust settle**”

Trigger flow if there is new raw data, and that data is at least 30 minutes old, it’s not already running, and no “in process” data is detected.

More Advanced Rule: “**trigger follow up flow**”

Trigger flow “ABCD_Follow_On” if there is NO new raw data and data has been processed more recently than the last time “ABCD_Follow_On” last ran, and it’s not already running, and no “in process” data is detected

More Advanced Rules: “**volume based triggering**”

Trigger flow “ABCD_Basic” if there is > N new raw data files and etc....

Trigger flow “ABCD_Full” if there is <= N new raw data and etc....

Further Basic Rules: “**Time Based**”

Trigger flow “ENV_Housekeeping” if it is N minutes since it was last triggered

Sequencing Rules: “**Time Based**”

Trigger flow “ABCD_Reporting” if ABCD data was processed more recently than the last trigger time of “ABCD_Reporting” and no “in process” data for ABCD is detected, and no “unprocessed” data for ABCD is detected

Caveats, notes and comments.

Some general points and caveats.

Using the LSF command line utilities requires:

“X commands” must be allowed

i.e. SAS needs to be allowed to call operating system commands

Process Manager needs to be configured for “single sign-on”

JS_LOGIN_REQUIRED=FALSE in js.conf file

Default install (with SAS) is TRUE for some reason

The condition(s) that are used in rules as criteria to cause a flow to trigger should be removed by that flow once it starts executing, e.g. the existence of unprocessed raw data goes away when the Ingest flow for that data progresses.

Triggering flows **uniquely** prevents “clashes”. You need to ensure that only one instance of a flow ever runs at a time, so when attempting to trigger a flow it is important to check that it is not already running.

Don’t mix and match this “rules based” technique with normal LSF time/file triggers, one principle this technique leverages is that it is in sole control of triggering the flows its managing.

You can achieve time based triggers using time slots and “every N minutes” rules, however, if simple time-based scheduling is required, just use the inbuilt LSF scheduling approach.

Consistent naming approach to jobs and flows - this approach doesn’t require that you have a consistent approach to naming jobs and flows, but the larger your catalogue of flows and jobs, the more benefits you’ll see from doing so.

So they tend to go hand in hand...

This “rules based” approach to scheduling opens up a range of possibilities, including “state machine” based ETL workload management, and “worker thread” and “queue based” ETL patterns.

In principle, this system could be build with alternative scheduling tools other than LSF.

FlowMaster– illustrative logic

Rules List

Select Active Rules

Check Time Slots

Check Not Running

Rules Engine

Flows to trigger

Trigger & Record

List of Running Flows

Raw Data Info

Flow History

SAS® GLOBAL FORUM 2016

IMAGINE. CREATE. INNOVATE.

LAS VEGAS | APRIL 18-21

#SASGF