syncsort

A Guide to DMX-h ETL Use Case Accelerators

Table of Contents

1 Introduction

DMX-h's design-once and deploy-everywhere methodology allows you to run the same application on a standalone machine, a Spark cluster, or a MapReduce cluster. Syncsort has provided a set of use case accelerators (UCAs), based on common use cases seen in the industry, to get you started on creating your own DMX-h solutions.

This guide presents an overview of:

- How to get the use case accelerators
- The use cases addressed
- The directory structure of the solutions and data
- The execution of the use cases both within and outside of the cluster

2 Getting the Use Case Accelerators

Syncsort provides the following pre-configured environments, where the DMX-h ETL software and UCAs are pre-installed:

- DMX-h ETL Test Drive VMs (register to get the desired VM download at www.syncsort.com/try)
  - Cloudera – see DMX-h Quick Start for Cloudera VM for more details
  - Hortonworks – see DMX-h Quick Start for Hortonworks Sandbox for more details
  - MapR – see DMX-h Quick Start for MapR VM for more details

If you are using one of these environments, the UCA examples are already set up for you on the edge node.

If you want to run the examples in your own cluster, you will need to do the following:

1. Download the zipped tar files DMX-h_UCA_Solutions.tar.gz and DMX-h_UCA_Data.tar.gz from the Guide to DMX-h Use Case Accelerators KB article. These contain the solutions and the sample data needed to run them.
2. Unzip and extract the downloaded files to the directory that you will designate as $DMXHADOOP_EXAMPLES_DIR on the Linux edge node in your cluster from which you will execute jobs, as follows:

```
cd <designated examples directory>
tar xvof DMX-h_UCA_Solutions.tar.gz
tar xvof DMX-h_UCA_Data.tar.gz
```

3 Summary of the Use Cases

The provided use case accelerators appear in three folders within the example directory structure:

- **DMXStandardJobs** – these are standard DMX-h ETL jobs that can be run on a single node or in the cluster, where they are automatically converted to Spark/MapReduce jobs as specified using Intelligent Execution (IX).
- **DMXUserDefinedMRJobs** – these are user-defined MapReduce jobs for cases when you want to specifically control how the job is run, though the IX solution is the recommended method. These can also be run either standalone or in the cluster.
- **DMExpress HDFS Load/Extract Jobs** – these are standard DMExpress ETL jobs run outside of the cluster for extracting HDFS and Mainframe data, and loading HDFS data.

The following sections further subcategorize the use case accelerators by function, describing each one and giving the name of the job sub-folder in which the files for executing that job will be found within the use case directory structure, described in section 4.

3.1 DMExpress Standard ETL Jobs

The following use case accelerators demonstrate common ETL scenarios and their corresponding DMX-h jobs which can be run either on a single node or in the cluster. Click on the linked name to go to the associated KB article. The corresponding jobname folder within the UCA directory structure described in section 4 is shown in [brackets].

3.1.1 Change Data Capture (CDC)

***CDC Single Output** [FileCDC]*

This use case accelerator demonstrates how to perform Change Data Capture (CDC), a process where previous and current versions of a large data set are compared to determine the changes that have occurred during the time period between the two versions. It joins two large files and produces a single output file that contains all changed records, with a flag added to each record indicating whether it is an insert, delete, or update.

***CDC Distributed Output** [FileCDC_MultiTargets]*

This use case accelerator is the same as FileCDC, except that it demonstrates how to produce three separate output files for the inserted, deleted, and updated records.

***Mainframe Extract + CDC** [MainframeCDCDemo]*

This use case accelerator is a combination of the Direct Mainframe Extract & Load and CDC Single Output accelerators.

3.1.2 Joins

Join Large Side with Small Side [FileJoinSmall]

This use case accelerator demonstrates how to perform an inner join between a small distributed cache file containing TPC-H supplier data and a large HDFS file containing lineitem data.

Join Large Side with Large Side [FileJoinLarge]

This use case accelerator demonstrates how to perform a join of two large files stored in HDFS. This example performs an inner join, but could easily be modified to perform a left-outer, right-outer or full-outer join.

3.1.3 Lookup

File Lookup [FileLookup]

This use case accelerator demonstrates how to perform a lookup in a small distributed cache file (TPC-H parts data) while processing a large HDFS file (lineitem data).

Lookup + Aggregation [FileLookupAggregation]

This use case accelerator demonstrates how to perform a lookup followed by an aggregation. Using TPC-H data, it performs a lookup in a TPC-H supplier data file based on the Supplier Key and retrieves the Supplier Account Balance; this value is then totaled for each supplier.

3.1.4 Aggregation

Web Log Aggregation [WebLogAggregation]

This use case accelerator demonstrates how to calculate the total number of visits per site in a set of web logs using aggregate tasks.

Word Count [WordCount]

This use case accelerator demonstrates how to perform the standard Hadoop word count example. It uses pivoting to tokenize the sentences into words, then aggregates on the words to obtain a total count for each.

3.1.5 Message Queues

Fraud Detection with Apache Kafka [FraudDetection]

This use case accelerator demonstrates how to read from and write to Apache Kafka message queue topics in a simplified fraud detection example, where non-fraudulent transactions are written to a Hive table. This UCA cannot be run in a MapR cluster.

3.2 DMExpress HDFS Load/Extract Jobs

The following use case accelerators demonstrate loading/extracting data to/from HDFS using DMExpress on the edge node. Click on the linked name to go to the associated KB article.

3.2.1 Mainframe Access and Integration

Direct Mainframe Extract & Load [MainframeExtractToHDFS]

This use case accelerator demonstrates how to load two files residing on a remote mainframe system to HDFS, converting the data from EBCDIC fixed length, mixed text and binary to ASCII displayable text in the process.

Mainframe File Load [MainframeExtractLocalToHDFS]

This use case accelerator demonstrates how to load two locally stored mainframe format files to HDFS, converting the data from EBCDIC fixed length, mixed text and binary to ASCII displayable text in the process.

Direct Mainframe Redefine Extract & Load [MainframeRedefineExtractToHDFS]

This use case accelerator demonstrates how to load a file residing on a remote mainframe system to HDFS, interpreting REDEFINES clauses and converting the data from EBCDIC fixed length, mixed text and binary to ASCII displayable text in the process.

Mainframe Redefine File Load [MainframeRedefineExtractLocalToHDFS]

This use case accelerator demonstrates how to load a locally stored mainframe format file to HDFS, interpreting REDEFINES clauses and converting the data from EBCDIC fixed length, mixed text and binary to ASCII displayable text in the process.

Mainframe Variable Processing [MainframeVariableProcessing]

This use case accelerator demonstrates how to load mainframe variable length EBCDIC-encoded files to HDFS, where they are converted into DMX-h's Mainframe Hadoop Distributable format so they can be subsequently processed in a distributed manner.

3.2.2 Connectivity

HDFS Extract [HDFSExtract]

This use case accelerator demonstrates how to extract TPC-H supplier data from the HDFS file system using HDFS connectivity in a DMExpress copy task.

HDFS Load [HDFSLoad]

This use case accelerator demonstrates how to load TPC-H supplier data to the HDFS file system, using HDFS connectivity in a DMExpress copy task.

HDFS Load Parallel [HDFSLoadParallel]

This use case accelerator is the same as HDFSLoad, except that the data is split into 5 partitions using the DMExpress random partition scheme and then loaded into HDFS in parallel.

4 Directory Structure of the Use Cases

The solutions and data are organized in the following directory structure, which should not be modified:

- $DMXHADOOP_EXAMPLES_DIR
  - bin [contains prep_dmx_example.sh]
  - Data [contains source files and a target folder for running outside of HDFS]
    - Source [contains sample source data used by the use case accelerators]
    - Target [target data directory for running examples outside of the cluster]
  - Jobs [contains example jobs]
    - <JobName> [one folder per example, such as FileCDC, WordCount, etc.]
      - DMXStandardJobs* [contains the jobs and tasks needed to run the standard version of the example for IX, including the prep_file_list required by prep_dmx_example.sh]
      - DMXUserDefinedMRJobs* [contains the jobs and tasks needed to run the user-defined MapReduce job, including the prep_file_list required by prep_dmx_example.sh]
        - MRJ_<JobName>.dxj [the MapReduce job]
        - J_<JobName>.dxj [the job which contains a MapReduce sub-job, if any]
        - MT_<MapTaskName>.dxt [a map-side task]
        - RT_<ReduceTaskName>.dxt [a reduce-side task, if any]
        - T_<TaskName>.dxt [a non-MapReduce task, if any]
      - DMXHDFSJobs* [contains DMExpress jobs for loading to and extracting from HDFS]

*These folders are only present where applicable for the given example. For UCAs that contain both DMXStandardJobs and DMXUserDefinedMRJobs folders, the recommended one to use is the IX solution given in DMXStandardJobs; the user-defined MapReduce solution is provided secondarily as a reference if there is a need to control the MapReduce flow in your own similar application.

5 Running the Use Case Accelerators

The use case accelerators can be run both within and outside of the cluster, via the Job Editor or the command line, in either a pre-configured DMX-h ETL environment or your own cluster.

For UCAs that have the `DMXStandardJobs` folder, which contains the IX solution, use the top-level job in that folder to run the UCA. For those that have only the `DMXUserDefinedMRJobs` folder, use the top-level job in that folder to run the UCA.

In all cases, some environment variables need to be set as defined in the following section, with more detailed instructions for each of the run cases in the subsequent sections.

Note that there are additional instructions in 5.3.2Appendix A for working with the Apache Kafka Fraud Detection UCA in DMExpress.

5.1 Setting the Environment Variables

The environment variables shown in the table below need to be defined as described.

When running in your own cluster, these variables need to be set manually. When running in a pre-configured DMX-h ETL environment, they are already set in the Hadoop-designated user's bash profile and the DMExpress Server dialog.

DMXHADOOP_EXAMPLES_DIR	Directory under which the solutions and data were extracted
HDFS_SERVER	Set to NameNode for running within the cluster, or to blank (or unset) for running outside of the cluster
HDFS_SOURCE_DIR[1]	Set to HDFS directory under which source files will be found (or copied to by the prep script); when testing Spark/MapReduce jobs outside of the cluster, set to local directory containing source files
HDFS_TARGET_DIR[1]	Set to HDFS directory under which target files will be written; when testing Spark/MapReduce jobs outside of the cluster, set to local directory in which to write target files
LOCAL_SOURCE_DIR[1]	Local directory containing source files; set to: `$DMXHADOOP_EXAMPLES_DIR/Data/Source`
LOCAL_TARGET_DIR[1]	Local directory under which target files will be written; can be set to: `$DMXHADOOP_EXAMPLES_DIR/Data/Target`
MAPRED_TEMP_DATA_DIR	Set to '.' for running within the cluster, or to the desired folder for intermediate data files when running outside of the cluster. This variable is needed for user-defined MapReduce only.
KAFKA_BIN_DIR[2]	Set to `<Kafka_home>/bin`
PATH[2]	Add `<DMExpress_home>/bin` to the path
LD_LIBRARY_PATH[2]	Add `<DMExpress_home>/lib` to the library path

ZOOKEEPER_SERVER[2,3]	Set to the hostname of the Zookeeper server.
DMX_CLASSPATH[2,3]	Set to the location of the jar files in the Kafka libs folder. For example, where `<Kafka_home>` is the Kafka installation folder: `export DMX_CLASSPATH="<Kafka_home>/libs/*.jar`
JAVA_HOME[3]	Set to the JRE installation directory. For example: `export JAVA_HOME=/usr/lib/jvm/java/jre/`
ODBCINI[3]	Set to the location of the odbc.ini file. For example: `export ODBCINI=/usr/dmexpress/etc/odbc.ini`
ODBCSYSINI[3]	Set to the directory containing the `odbcinst.ini` file. For example: `export ODBCSYSINI=/usr/dmexpress/etc`
CLOUDERAHIVEINI[3]	For CDH clusters, set to the location of the `cloudera.hiveodbc.ini` file. For example: `export CLOUDERAHIVEINI=/usr/dmexpress/etc/cloudera.hiveodbc.ini`
HORTONWORKSHIVEINI[3]	For HDP clusters, set to the location of the `hortonworks.hiveodbc.ini` file. For example: `export HORTONWORKSHIVEINI=/usr/dmexpress/etc/hortonworks.hiveodbc.ini`
DMX_HADOOP_ON_VALIDATION_FAILURE[3]	Set to `LOCALNODE`. For example: `export DMX_HADOOP_ON_VALIDATION_FAILURE=LOCALNODE`

[1] To enable data sampling at development time, set LOCAL_SOURCE_DIR, LOCAL_TARGET_DIR, HDFS_SOURCE_DIR, and HDFS_TARGET_DIR, as needed, in the DMExpress Server dialog. Prefix the path with the Linux server name to sample files on the Linux edge node, e.g. myserver:/mypath.

[2] Needed only for running the prep script to load the Apache Kafka Fraud Detection UCA data. If the prep script is run for all UCAs and any of these variables are not set, the load of the Kafka data will fail, but all other UCA data will be loaded successfully.

[3] Needed only for running the Apache Kafka Fraud Detection UCA.

5.2 Running within the Cluster

Prior to running the use cases, it is necessary to prepare the HDFS environment. The use case accelerators can then be run from either the Job Editor or the command line.

5.2.1 Running the Prep Script to Prepare the HDFS Environment

If the sample data needed to run the job is not already loaded into HDFS, you'll need to run the `$DMXHADOOP_EXAMPLES_DIR/bin/prep_dmx_example.sh` script. If

running in your own cluster, set the following environment variables in the command line environment as indicated in the table above:

- DMXHADOOP_EXAMPLES_DIR
- HDFS_SOURCE_DIR
- HDFS_TARGET_DIR
- LOCAL_SOURCE_DIR

Then run the script in one of the following ways to load the data for all the examples or for specific ones:

```
prep_dmx_example.sh ALL
prep_dmx_example.sh <space-separated list of example folder
   names>
```

This script uses the prep_file_list in the example's DMXStandardJobs or DMXUserDefinedMRJobs folder to load the specified list of sample data files to HDFS.

5.2.2 Running the Examples within the Spark/MapReduce Cluster via the Job Editor

The following steps describe how to run the use case accelerators within the cluster from the Job Editor:

1. Open the Job Editor on a Windows machine.
2. Using a Remote Server File Browsing connection, connect to your cluster's edge/ETL node or the Test Drive VM.
   a. For your own cluster, select FTP or Secure FTP connection and specify the corresponding login credentials as required.
   b. For the pre-configured VMs, specify the connection as follows, where `<distribution>` is `cdh` (Cloudera), `hdp` (Hortonworks), or `mapr` (MapR):

      Server: `dmxhtstdrv<distribution>`

      Connection type: Secure FTP

      Authentication: Password

      User name: `dmxdemo`

      Password: `dmxd3mo`
3. Browse to the location and select the `J_<JobName>.dxj` version of the use case accelerator job you want to run.
   a. If running in your own cluster, define the environment variables (described in section 0) in the **Environment Variables** tab of the **DMExpress Server** dialog, accessed via **Run->Define environment variables**. Be sure to first select the Linux server on which you will run the job, then enter the environment variables. If running in a pre-configured environment, these are picked up from the pre-configured user profile.
4. Click on the **Run** button to bring up the **Run Job** dialog.

a. If not already pointing to the correct server, select the server on which you want to run the job, and enter the required credentials as specified in step 2 above).

b. In the **Runon** section of the **Run Job** dialog, specify the framework in which to run the job:

i. To run in Spark, select **Spark** for the **Framework**, click on **Define** for the **Spark master URL**, select **yarn** for the **Spark cluster type**, click **OK** to dismiss the **Spark Master URL** dialog, then click **OK** in the **Run job** dialog.

ii. To run in MapReduce, select **MapReduce** for the **Framework**, and click **OK**.

c.

5. This will bring up the **DMExpress Server** dialog, which will show the progress of the running job. Upon completion, select the job and click on **Detail…** to see messages and statistics. (The `SRVCDFL` warning message about the data directory can be safely ignored.)

a. To view the MapReduce logs, see Where to Find DMExpress Hadoop Logs on MRv1 or Where to Find DMExpress Hadoop Logs on YARN (MRv2).

b. Spark standalone and Spark on YARN logs can be found via the Spark History Server Web UI at `http://<hostname>:<spark_web_ui_port>`, or via the Spark link in Cloudera Manager. Spark on YARN logs can also be found via the Hadoop JobHistoryServer.

5.2.3 Running the Examples within the Spark/MapReduce Cluster via the Command Line

The following steps describe how to run the use case accelerators within the cluster from the command line:

1. If running in your own cluster, set the environment variables as described in section 0.

2. Run one of the following commands, where the `/RUNON` option specifies whether to run in Spark or MapReduce, `<JobType>` is `DMXStandardJobs` for the IX-defined UCAs (and `DMXUserDefinedMRJobs` for the non-IX-defined UCAs), and the `/EXPORT` option is used to pass through any environment variables from the table in section 0 that were set to run this job:

```
dmxjob /RUN
   $DMXHADOOP_EXAMPLES_DIR/Jobs/<JobName>/<JobType>/<JobNam
   e>.dxj /RUNON MAPREDUCE /EXPORT <space-separated list of
   environment variable names>

dmxjob /RUN
   $DMXHADOOP_EXAMPLES_DIR/Jobs/<JobName>/<JobType>/<JobNam
   e>.dxj /RUNON SPARK yarn /EXPORT <space-separated list
   of environment variable names>
```

See “Running a job from the command prompt” in the DMExpress Help for details on other Spark options.

5.3 Running Outside of the Cluster

The use case accelerators can be run outside of the cluster for testing from either the Job Editor or the command line. Additionally, the HDFS load and extract examples are run outside of the cluster as regular DMExpress jobs.

5.3.1 Running the Examples outside of the cluster via the Job Editor

Run the use case accelerators as described in section 5.2.2, with the following adjustments:

1. If running in your own cluster, or if testing Spark/MapReduce jobs in a pre-configured environment, where the environment variables are pre-set for running in the cluster, set the environment variables as described in section 0 for running outside of the cluster.
2. Select **Local** for the **Framework** in the **Runon** section of the **Run Job** dialog.

5.3.2 Running the Examples outside of the cluster via the Command Line

The following steps describe how to run the use case accelerators outside of the cluster from the command line:

1. If running in your own cluster, or if testing Spark/MapReduce jobs in a pre-configured environment, where the environment variables are pre-set for running in the cluster, set the environment variables as described in section 0 for running outside of the cluster.
2. Execute the following command, where `<JobType>` is `DMXStandardJobs` for the IX-defined UCAs and `DMXUserDefinedMRJobs` for the non-IX-defined UCAs:

```
dmxjob /RUN
   $DMXHADOOP_EXAMPLES_DIR/Jobs/<JobName>/<JobType>/<JobNam
   e>.dxj
```

Appendix A Additional Instructions for the Apache Kafka Fraud Detection UCA

The following steps need to be taken on your Windows machine in order to modify or run the Apache Kafka Fraud Detection UCA from the DMExpress GUI.

A.1 Kafka

- Download the Windows 32-bit Kafka libraries (see “Platform, package and feature support matrix” in the DMExpress Help for the currently supported version of Apache Kafka) from kafka.apache.org/downloads.html, and extract the download on the Windows machine where the Kafka connection will be made.
- Set the following Windows System environment variables via the Control Panel:
  - DMX_CLASSPATH must be set to point to the location of all the jar files in the extracted Kafka libs folder, for example:

    ```
    set DMX_CLASSPATH="C:\Program Files(x86)\kafka_2.11-0.9.0.0\libs\*.jar"
    ```

  - JAVA_HOME must be set to point to your JDK or JRE installation, for example:

    ```
    set JAVA_HOME="C:\Program Files (x86)\Java\jdk1.7.0_51\jre"
    ```

A.2 Hive ODBC

- Download the Windows 32-bit Hive ODBC driver and associated documentation from one of the following Hadoop vendor websites:
  - Cloudera
  - Hortonworks
- Install the downloaded ODBC driver.
- Start the **ODBC Data Source Administrator** by following the instructions for Windows systems in “Defining ODBC data sources” in the DMExpress Help.
- In the **Create New Data Source** dialog, select your Hive ODBC vendor's driver from the list, and then click **Finish**.
- Use the following settings in the **ODBC Driver DSN Setup** dialog for a Hive ODBC data source:
  - **Data Source Name**: Hive ODBC
  - **Host**: `dmxhtstdrvcdh` or `dmxhtstdrvhdp`
  - **Port**: 10000
  - **Database**: default
  - **Hive Server Type**: Hive Server 2
  - **Authentication Mechanism**: User Name
  - **Advanced Options**: Check **Use Native Query** and click **OK**. Some ODBC Hive drivers work with both HiveQL and SQL query languages. This setting enables the user of native HiveQL instead of SQL

About Syncsort

Syncsort provides enterprise software that allows organizations to collect, integrate, sort, and distribute more data in less time, with fewer resources and lower costs. Thousands of customers in more than 85 countries, including 87 of the Fortune 100 companies, use our fast and secure software to optimize and offload data processing workloads. Powering over 50% of the world's mainframes, Syncsort software provides specialized solutions spanning "Big Iron to Big Data", including next gen analytical platforms such as Hadoop, cloud, and Splunk. For more than 40 years, customers have turned to Syncsort's software and expertise to dramatically improve performance of their data processing environments, while reducing hardware and labor costs. Experience Syncsort at www.syncsort.com.

Syncsort Inc. ***50 Tice Boulevard, Suite 250, Woodcliff Lake, NJ 07677*** ***201.930.8200***