INTEGRATING BIG DATA... SMARTER

syncsort

Creating a DMX-h Job

Tutorial

Table of Contents

1 Introduction

DMX-h ETL is Syncsort's high-performance ETL software for Hadoop. It combines powerful ETL data processing capabilities with the enhanced performance and scalability of Hadoop, without the need to learn complex MapReduce programming skills.

Once you have DMX-h properly installed in your cluster and on a Windows machine, you're ready to start creating MapReduce jobs. This tutorial will show you how easy it is to create an aggregation MapReduce job in DMX-h.

For assistance with DMX-h ETL, please visit the DMX-h ETL group in the Syncsort User Community.

2 The Development Environment

After installing the DMExpress Graphical User Interface on your Windows Workstation, you will find the components you need to develop a DMX-h MapReduce job under the **Start->DMExpress** menu.

2.1 DMExpress Job Editor

The Job Editor allows you set up your overall data flow. Here you will be able to add jobs and tasks to the canvas and create sequences between them to develop, manage, and oversee the data flow from your sources to your targets.

2.2 DMExpress Task Editor

The Task Editor is where you define your transformations. DMX-h will allow you to perform set level transformations (Join, Copy, Merge, Sort, or Aggregation) on your data set. From here, you can also specify how to perform transformations on a field level basis (Date/time functions, string manipulations, lookups, mathematical operations, etc.)

2.3 DMExpress Server

The Server Status Dialog will allow you to connect to any DMExpress Server or local workstation to view scheduled jobs, monitor job execution status, or modify DMExpress-specific environment variables.

3 Creating a DMX-h Job

In this example, we will be taking LineItem order data and aggregating it to figure out how much money is made per shipping method.

3.1 Setting up the Windows Environment

Before we begin development, we will prepare our workstation by creating directories to save our Jobs and Data. For this example, we will use “C:\DMX\” as the root location, but you can choose any path you’d like.

- Create a directory for the MapReduce Job and Task files

  `C:\DMX\DMXJobs\MapReduce\MyFirstJob\`

- Create a directory for the MapReduce Input Data and copy the provided sample data to it

  `C:\DMX\DMXData\MapReduce\input\MyFirstJob`

- Create a directory for the MapReduce Output data

  `C:\DMX\DMXData\MapReduce\output\MyFirstJob`

3.2 Opening the Job Editor

When you start the Job Editor, the first thing you want to do is specify the directory where DMExpress can find your source and target data. In this example, we are going to specify that it be relative to our current path. To do this, select **Edit**->**Job Settings** and specify the **Runtime data directory** as a single dot, which means the current directory. This will set the environment variable `DMXDataDirectory`, which is used in several places in the job to specify file locations.

You can now save the job by selecting **File->Save Job as…**, navigating to the directory you created for jobs earlier (`C:\DMX\DMXJobs\MapReduce\MyFirstJob\`), and saving it there as `MapReduceJob.dxj`. The file extension `.dxj` indicates a DMX job.

There are other environment variables we can set to make job development easier. From the Job Editor, select **Run->Define Environment Variables…** and set the following variables in the table:

- `DMX_HADOOP_NUM_REDUCERS = 1`
- `HDFS_SERVER =` (leave this blank for now)
- `HDFS_SOURCE_DIR = C:\DMX\DMXData\MapReduce\input`
- `HDFS_TARGET_DIR = C:\DMX\DMXData\MapReduce\output`
- `OUTPUT = part-00000`
- `TEMP_DATA_DIR = C:\tmp` (you can change this to whatever you want)

Since we are setting the number of reducers to 1, we can just set the single output file to part-00000, using the standard Hadoop file name format for reducer output.

3.3 Creating the Mapper

To create the mapper task, select **Edit**->**Add New DMExpress Task…**. Specify `Mapper.dxt` as the name of the task and save it in the same directory where you saved the job (`C:\DMX\DMXJobs\MapReduce\MyFirstJob\`). We now have the task added to our canvas and can open it directly from there by double clicking on the task.

We are now in the Task Editor. There are six main components to task development that we'll address in turn as we go through the tutorial:

1. Selecting the Task Type
2. Defining the source of your data
3. Defining the layout of that source
4. Defining the parameters of your set level transformation
5. Defining the target for your transformed data
6. Defining how the target data should be reformatted

3.3.1 Selecting the Task Type

DMExpress uses a template approach to job development. The Task Editor window contains a task tree that defines the components needed to define the task based on the selected task type – Aggregate, Copy, Join, Merge, and Sort. For this example, click on the **Aggregate** button in the toolbar to create an Aggregate task.

3.3.2 Defining the Source

Now we need to define the source of the data. Right click on **Source** in the task tree, and select **Add File…**. Since this will be a MapReduce job, we will specify an HDFS connection for our data. To do this, click on the **Add new…** button next to the **Remote file connection**, fill in the required information as follows, and click **OK**:

- **Name:** `HDFSConnection`
- **File server:** `$HDFS_SERVER`
- **Current DMExpress server:** *`the name of your windows workstation`*
- **Authentication: None**

There are a few things to note here. By using the environment variable `$HDFS_SERVER` as our connection, we have the flexibility to deploy this job to multiple environments without needing to change any of the tasks. It also gives us the ability to run this job completely locally on our windows machine without the need to connect to our Hadoop cluster. When we set our environment variables, we left this blank. When the connection is blank, it falls back to running on the local file system, which is what we want to do during our development.

For our file name, we will again be using environment variables so that we can run the job in multiple locations. Specify the filename as `$HDFS_SOURCE_DIR/MyFirstJob/Sample_LineItems.txt`. Since we set `$HDFS_SOURCE_DIR` to our input directory in the previous step, it will link to the `Sample_LineItems.txt` file we copied there.

Our file is LF terminated, so choose **Text, Unix (LF record terminator)** for **File type** and click **OK, leaving** the rest of the options with their default settings.

3.3.3 Defining the Source Layout

Now that we have a source file, we need to tell DMX-h how to interpret the fields in the file by defining the source layout. Right click on the file you just added and select **Map Record**. This will bring up 100 sampled records from the file so we can easily define the fields.

For this example, we don't need to define all the fields, since we will not need them in our output. Change the field name of **Field6** to **ExtendedPrice** and **Field14** to **ShippingMethod**, and change the data type of **ExtendedPrice** to **Number**. Click **OK**.

3.3.4 Defining the Partitioning Scheme

An important step in the development of the mapper task is creating a partitioning scheme to direct the data from the various mappers to the appropriate reducer to ensure we get the correct aggregation results for each shipping method. To do this, we will create a stored **Value** called `PartitionKey` which will do a CRC32 hash on the shipping method, using the number of reducers as the divisor.

Select **Metadata->Add Value…** to bring up the **Expression Builder** dialog. From here, you can create all kinds of field level transformations on your data. Change the **Name** field to `PartitionKey`.

To create the expression, select **Hash**->**CRC32** from the **Function** dropdown. This will pre-populate the expression with the parameters it requires and auto-highlight the first field.

Double click the `ShippingMethod` field that we defined on the left hand side, and it will replace `value` with `ShippingMethod`. Now click on the `divisor` field to highlight the whole word. Replace that with the environment variable we defined earlier to represent the number of reducers and convert it to a number using the built in `ToNumber()` function. Environment variables must be enclosed in double quotes, so the completed expression should look as follows:

```
CRC32(recordlayout1.ShippingMethod,
   ToNumber("$DMX_HADOOP_NUM_REDUCERS"))
```

Click **OK**.

3.3.5 Defining the Aggregation Criteria

Double click on **Aggregate** in the task tree. A dialog box will pop up showing the source fields and values available to **Group by** and **Summarize**. Select `ShippingMethod` on the left and click the yellow arrow to group by that field. Select `ExtendedPrice` on the left and click the blue arrow to summarize that field. Click the dropdown in the **Function** field and select **Total** to get the sum of `ExtendedPrice` for the `ShippingMethod`.

Since we are using DMX-h to replace the map sort step, we need to sort by the partition key we created earlier to send all records with a given partition key to the same reducer. To do this, click on `PartitionKey` on the left and click the yellow arrow to add it to the **Group by** list. To move `PartitionKey` into the first position so that we sort first on that and then on `ShippingMethod`, select it and click on the up arrow in the top right corner of the **Group by** list. Finally, check the box for **Sort aggregated records on the group by fields for target** and click **OK**.

3.3.6 Defining the Target

The target we define here will be an intermediate placeholder file between the mapper and the reducer. In fact, when we run this in Hadoop, the intermediate file won't even be created, as DMX-h will automatically convert the file to a stream, working seamlessly within MapReduce.

Right click on **Target** in the task tree and select **Add File…**. This will bring up a dialog box similar to when you added the source file. This is a temporary file for the map output, so name it `$TEMP_DATA_DIR/MyFirstJob_MapOutput.txt`. Ensure that the **File type** is set as **Text, UNIX (LF record terminator)** and that the radio button for **When file exists** is set to **Overwrite file**. Click **OK**.

3.3.7 Defining the Reformat

The reformat defines the output layout. To create a reformat for our target file, right click on the target file and select **Add Reformat…**. A dialog box will pop up that allows us to define which fields we want to output and perform any field level transformations we choose.

For DMX-h to properly recognize the partition key, it must be the first field of the mapper output, so click on `PartitionKey` on the left and click the green arrow to add it to the output. The `ShippingMethod` field will become auto-selected, so clicking the green arrow again will bring that field to the target as well. Click it one more time to add the `ExtendedPrice` field to the target. You can rename any of the fields by clicking in the corresponding cell for the **Target field name**. Change `total_ExtendedPrice` to `SubTotalExtendedPrice`, and click **OK**.

We're now done with the mapper, so we can save the task and close the Task Editor. This will return us back to the Job Editor, with the mapper task defined on the canvas as an Aggregation. If you right click on the task and select **Expand**, you can see the details of the task.

3.4 Creating the Reducer

The reducer step will merge all the outputs from the various mappers, sum the subtotals for each shipping method, and give us our final sum.

To create the reducer task, select **Edit**->**Add New DMExpress Task…**. Specify `Reducer.dxt` as the name of the task and save it in the same directory where you saved the job (`C:\DMX\DMXJobs\MapReduce\MyFirstJob\`). We now have the task added to our canvas and can open it directly from there by double clicking on the task.

3.4.1 Selecting the Task Type

Since the reducer will be aggregating the subtotals from the mappers, again click on the **Aggregate** button in the toolbar to create an Aggregate task.

3.4.2 Adding the Source

The source for this task will be the same as the output from our mapper. Right click on **Source** in the task tree and select **Add File…**. Set the **File name** to `$TEMP_DATA_DIR/MyFirstJob_MapOutput.txt`, the **File type** to **Text, Unix (LF record Terminator)**, and click **OK**.

3.4.3 Adding External Metadata

A useful feature of DMExpress is the ability to reference metadata created in other tasks. Here, we will use the target reformat from the mapper as the source layout for the reducer.

Right click on **Metadata** in the task tree, select **Link to External Metadata...**, and browse to the previous task, `Mapper.dxt`. DMExpress populates the dialog with all the available metadata elements from the mapper task.

The two elements of interest to us are `recordlayout_MyFirstJob_MapOutput_1`, which we'll use as our source layout, and `HDFSConnection`, which we'll use when defining the output of our reducer. Clear the checkbox for the remaining elements (`recordlayout1` and `PartitionKey`), which are not needed in the task, and click **OK**. Click **Yes** when prompted to automatically link the source layout we are importing to our source.

3.4.4 Defining the Aggregation Criteria

Double click on **Aggregate** in the task tree. Since we are in the reducer, we no longer care about the partition key. Select `ShippingMethod` on the left and click the yellow arrow to add it to the **Group by** list. Select `SubtotalExtendedPrice` on the left and click the blue arrow to add it to the **Summarize** list, which defaults to **Total**. Click **OK**.

3.4.5 Defining the Target

Now we want to define where the output of our MapReduce job will go. Right click on **Target** in the task tree, select **Add File…**, and change the **File name** to `$HDFS_TARGET_DIR/MyFirstJob/$OUTPUT`.

Since we have already imported the **Remote file connection** information from the mapper, we can simply click on the drop down and select `HDFSConnection`.

Ensure that the **File type** is **Text, Unix (LF record terminator)** and that **Overwrite file** is selected, and click **OK**.

3.4.6 Defining the Reformat

Right click on the target file you created, select **Add Reformat…**, and add both fields from the source to the reformat. Change the name of `total_SubtotalExtendedPrice` to `TotalExtendedPrice` and click **OK**.

3.5 Finalizing the job

We're now done with development of the mapper and reducer. Save the task, close the Task Editor, and return to the Job Editor.

Right click on the reducer task and select **Expand** to see the full details of the job. You'll notice that a sequence arrow was automatically added between the mapper and reducer since the output file name/location of the mapper matches the input file name/location of the reducer. Right click on this arrow and select **Use MapReduce Sequence** to specify that this is the separator between the mapper and reducer.

4 Running the Job Locally

Now that we are done with development of our MapReduce job, we will be able to run it locally on our Windows workstation without a connection to any Hadoop cluster. This is extremely helpful for agile development and debugging in an isolated environment.

4.1 Sampling the Source Data

Another helpful feature of DMX-h is the ability to sample your source data directly from the job editor. If you right click on the Sample_LineItems.txt file in the Mapper task and select **Sample…**, you will be able to see the first 100 records in the file. Click Close after you're done familiarizing yourself with the source.

4.2 Running the Job Locally

Now we are ready to run the job as a standalone ETL job on our Windows workstation. Click on the **Run** button in the toolbar. A dialog box will appear allowing you to set various options for the execution of your job. No changes to the defaults are required, so click OK to begin execution.

The Run Dialog will automatically close and bring up the Server Status dialog. From here you can monitor previously executed jobs, currently running jobs, and future scheduled jobs. You will be able to see the job you are running execute and complete. You can double click on the job in the status to see the execution details. You can see things such as execution time, records read, records aggregated, records output, machine utilization, and so on. Close the dialogs once done.

4.3 Sampling the Results

Now that that job has completed, you can sample the intermediate output (MyFirstJob_MapOutput.txt) and the final results ($OUTPUT). This is helpful to verify that the functionality developed is correct before ever deploying to a Hadoop cluster. This will save development time by cutting out multiple deployments, and it will save Hadoop cluster resources.

5 Running the Job in Hadoop

5.1 Setting up the Hadoop Environment

Now that we have completed development and debugging, the next step is to prepare the Hadoop cluster to deploy and execute the job in MapReduce.

5.1.1 Creating Linux Local Directories

Connect to an execution node (either an edge node, the Job Tracker, or the NameNode) on your Hadoop cluster via a shell.

Create a directory for your jobs and tasks within your home directory.

```
$ mkdir ~/MapReduce/MyFirstJob
```

5.1.2 Creating HDFS Directories for the Sample Data

Create HDFS directories for your job input and output.

```
$ hadoop fs -mkdir input/MyFirstJob

$ hadoop fs -mkdir output
```

This will create directories for the user you are logged in as. The full path is important as we will need to define it later in our Linux environment variables. It will typically be something like `/user/`*`username`*`/input/MyFirstJob`.

5.1.3 Copying Sample Data to HDFS

Now we are ready to load the data into HDFS. First, transfer the sample file from your Windows workstation to the `~/MapReduce/MyFirstJob/` directory on your execution node using FTP, SCP, etc. For example, using SCP from Cygwin:

```
$ scp
   /cygdrive/c/DMX/DMXData/MapReduce/input/MyFirstJob/Sample_LineItem
   s.txt user@execution_node:/home/user/MapReduce/MyFirstJob/
```

Then log into the execution node using SSH or a similar protocol and load the data into HDFS as follows:

```
$ hadoop fs -put ~/MapReduce/MyFirstJob/Sample_LineItems.txt
   input/MyFirstJob/
```

5.1.4 Saving the Job to the Newly Created Job Directories

Deployment of the job is easy to do from within the job editor. Select **File->Save Job As…**, select the **Remote Servers** tab, and double click on **New file browsing connection**.

Fill in the information to connect via FTP or SFTP to the execution node from which you will be running DMX-h MapReduce Jobs, click on **Verify connection**, click on **OK**, then **OK** again. Browse to `/home/`*`user`*`/MapReduce/MyFirstJob` and click **Save**.

The job and tasks have now been saved to your Hadoop execution node.

5.1.5 Changing the Server to the Edge Node

We can now connect to the DMX Server running on the execution node. Click on the **Status** button in the toolbar. In the **DMExpress Server Connection** dialog, select the **UNIX** tab, specify the Linux execution node connection information, and click **OK**.

5.1.6 Setting the Environment Variables for Hadoop

To run in Hadoop instead of Windows, we need to change the environment variable values. In the **DMExpress Server** dialog, select the **Environment Variables** tab and modify the environment variables as follows:

- `DMX_HADOOP_MAPRED_REDUCE_TASKS = 1` (this sets the value of `DMX_HADOOP_NUM_REDUCERS` at run time)
- `HDFS_SERVER` = *`machine name of your HDFS NameNode`*
- `HDFS_SOURCE_DIR = /user/`*`username`*`/input`
- `HDFS_TARGET_DIR = /user/`*`username`*`/output`
- `TEMP_DATA_DIR = .` (the current working directory)

Click **Close**.

5.2 Running as a Hadoop Job

Click on the **Run** button in the toolbar, change the **Run on** setting to **Hadoop cluster**, leave the remaining defaults, and click **OK**. The server was already set to point to the execution node when we modified the environment variables, so the job will be invoked on the execution node and sent to the Hadoop cluster to run in MapReduce.

You can monitor directly from the **Server Status** dialog just like we did for the Windows execution. Alternatively, you can also navigate to the Job Tracker site (http://<JobTrackerNodeName>:50030/) to view the status there.

About Syncsort

Syncsort provides data-intensive organizations across the big data continuum with a smarter way to collect and process the ever expanding data avalanche. With thousands of deployments across all major platforms, including mainframe, Syncsort helps customers around the world overcome the architectural limits of today's data integration and Hadoop environments, empowering their organizations to drive better business outcomes, in less time – with fewer resources and lower total cost of ownership. For more information, please visit www.syncsort.com.

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