Introduction to different concepts in Mapping Dataflows

Last updated by | Ranjith Katukojwala | Mar 2, 2023 at 4:21 PM PST

Contents

What are ADF (Azure Data Factory) Mapping Dataflows?

Mapping Dataflows or in short Dataflows are visual tools based on sparks to do data processing, be it small or big data. UX (User Experience) and backend get user intent in form of DSL (dataflow script language), users do not need to learn any of this language. Dataflow has lots of transformations and is in an active development phase to support different customer scenarios.

Where is data being processed in the case of Dataflows?

Data is processed on spark clusters, which are created on demand. These clusters are compliant with enterprise level security and compliance. Every job runs in isolation on a cluster, so that consumers get the same performance every time. Clusters are created on demand as required and deleted once the job is done, unless

specified with TTL (Time to Live) to be reused to save creation time for follow-up jobs. Dataflow clusters are not shared among different jobs and tenants.

How much data can be processed with Dataflow?

Dataflows is a big data solution. Dataflows use spark internally which will add more data disks as required on spark clusters to work with huge data which cannot fit in the cluster. Although small clusters can also work on big data, but to finish the work faster, an appropriate cluster size is better to use, as bigger clusters tend to finish the job faster. In case sources and sinks can handle fast read/write bigger clusters can finish jobs faster.

Does Dataflows have value over spark?

Dataflows use spark internally but provide more value to customers. For example

- No clusters to be managed
- No code to be maintained, if your expertise is to shape the data, then use that without coding.
- Many transformations which are not available directly in Spark or required sophisticated coding to achieve scenarios
- Compliance like no secrets to be stored or used in code.
- Features like reading in parallel from SQL sources, staging data for sources like Synapse, pre/post SQLs (Structured Query Language), pre/post file commands etc.
- Other features like schema drift, column patterns, rank over whole dataset etc.

How do Dataflows run internally?

Dataflows internally use spark, but it offers other values like handling the wild card path, ignoring/throwing error on empty source files, staging the data from Synapse, schema drift etc.

While execution each sink is processed separately (unless "Run in parallel" option is enabled), and we start analyzing from sink, going back to every source utilized in the sink. Each run can be divided into pre-processing, spark job and post-processing. Pre-processing includes preparing the source data or knowing the source schema etc. Spark job is reading the source data directly from source or from staging area, transforming the data, and writing in to sink or staging area. Post-processing involves moving the data from staging to target, or putting it in specific files, running post operations like post SQL, moving files after jobs etc.

Following are the few scenarios which are explained on how dataflows work.

Schema drift and pattern match: Dataflows first assess the source data for schema. It samples the data to find the schema for the run. Schema drift works by knowing the schema before the run comparing it with schema specified in the source transformation. All the patterns are being matched to create the spark job.

Staging for Synapse and hive sources/sinks: Hive and Synapse sources/sinks support staging, all the staging for data is done before the spark job starts. So usually, you will see some time being spent on source initialization or staging.

Wild card path for files: File based sources support different reading options like wild card path, list of files, ignore or fail with no data found with pattern, modified dates for files etc. These options have different handling and are preprocessed before spark job start. Like for wild card paths all the files in source folders are being assessed, as you can realize this will be a costly operation as files are compared against the wild card

path. Dataflows add additional value for wild card path by giving option to ignore/fail in case of no file match, as well as nested folders. This handling comes with additional calls to storage layers and hence results in time consumption. Once the file list is prepared the spark job is configured and run.

Single file on sink: In general spark write the data in multiple partitioned files. Dataflow offers writing to single file, this is done by reading the partitioned files and writing to a single file in single thread fashion. Writing a single file cannot be done in parallel. Dataflow does not use spark to coalesce to single partition as that works only for small data sizes which can fit in memory. This option impacts the performance of Dataflow a lot.

Writing to file as per column data: Dataflows offers writing rows to the files names as mentioned in columns. There are multiple uses of this, like routing the rows to same file from where it is read when used with similar option to store the file name in column for each row. Writing to a specific file requires partitioned data to be merged into single file like single file option with a difference that there are many such files. So, this option has a significant performance impact. If rather than writing to file, writing to folder can be used that will be more performant option with similar concept of dividing the data into different folders as per data specifying folder name in the files.

Dataflow and parallel processing: Dataflows process a lot of data in parallel. Like for a file source multiple files are read in parallel creating different partitions. Every 128 MBs data size will create a partition in case of file-based sources, similar techniques are there for other kind of sources, like for SQL we offer Source based partitioning scheme. Based on how many cores are available many of these partitions will be active all the time. Many operations are performed on these partitions in parallel. There are many transformations which do not need to look across different rows and those are performed together on every row in a partition. Examples of these transformations are Derived Column, Parse, Select, Stringify, Union, Call etc. There are other sets of transformations which required whole dataset to look in to, like where we do group by like in aggregate, window, pivot, unpivot etc.; apart from these other examples are joins, lookup, exists where one of the streams is compared with all rows from other stream. In such transformations data is staged to disks and you can see it in monitoring that a stage is created for Join or such transformations along with other row level transformations bundled together.

What is run in parallel option: In Dataflow every sink is processed in order, and data for that sink is being fetched and transformed from various sources. In some scenarios like where sink has some extra processing (on driver node) or spark job itself does not require full cluster, then "Run in parallel" option can be used. "Run in parallel" enables multiple sinks to be processed in parallel which have same sink order. This option helps spark clusters to be utilized fully in case it is not being used. There are usually a few scenarios which can benefit from this option. A user should test this option with production data and see if this option benefits.

Using disks in case of more data: In case there is no space on cluster to process enormous amounts of the data then disks are added on fly to complete the job. The amount of space required to run the job depends on input data, number of transformations, partitioning optimizations, number of sinks. If source/sinks can work with satisfactory performance with larger clusters, then it is better to choose larger clusters to avoid additional disks. In case source is SQL which cannot push data out with large throughput it is better to keep smaller cluster, although disks will be added later, but there will be saving on number of cores, as cores will not be utilized in case data is not coming with good speed. Systems like files, cosmos DB etc. are fast to make use of larger clusters.

How are rows interested/deleted/updated to target table?Dataflow reads all the files together from the source datasets (not file by file) and based on the logic 4 different sets of rows are prepared to delete, update, insert and upserts. After loading the data in to temp table on table sinks data, data is loaded to target tables using

SQL statements (1 statement each if enabled for delete, insert, update, upsert). If error row handling is enabled, then based on settings multiple batches will be used.

How to determine cluster size?

Dataflow uses spark which can run any amount of data through any size of cluster (there are exceptions), but you can use any kind of cluster, for big data scenarios it is good to put the decent size cluster, like 32+16 cores. In case clusters are small for data, managed disks are added to the cluster to handle big data, which cannot fit in cluster while processing. After dataflow is created, run the dataflow as part of pipeline with production size data to find out if with a particular number of cores dataflow is completing in time you are looking for, otherwise adjust the cores.

How to develop a Dataflow?

Dataflow provides intuitive UX to add transformation, where you can also do the data preview on each step. All transformations have documentation link in the UX to learn more as required. All transformation has Inspect and Data preview tab. Inspect shows how many columns were created or used in current transformation. Dataflow offers Expression Builder to write expressions. Expression builder provides data preview for current expressions, auto completion, error marking, help of functions etc.

Dataflow UX also keeps track to available columns at a particular point, although these columns should be defined either in projection tab in source or other transformations. Drifted columns can be visible along with other data in Data Preview. Once logic is good for Dataflow, Dataflow can be further tested as part of pipeline along with other activities to test the overall flow of pipeline.

Dataflow support parameterization like the rest of ADF which can be tested using debug pipeline runs before publishing the pipelines and dataflows.

What are Flowlets?

Flowlets are built to create reusable components of Dataflows. For example, write a flowlet which can clean the address column, now this flowlet can be used in different Dataflows. Flowlets increase productivity by creating reusable components. Flowlets can interact with sources, supports parameters, all transformations, and can be called from other flowlets. Flowlets are used by reference, any change in flowlet results in logic update in all the dataflows using it.

How to monitor dataflows?

Dataflows can be monitored from UX. There is good amount of information on various stages and pre/post steps in the monitoring view.

Like clicking on a source can provide time for staging and file system initialization time

Clicking on the sink gives information about the post steps and additional details

What is broadcast in joins?

Joins/lookup/exist transformations work by comparing rows from one stream with rows from another stream. This means if Dataflow is working on rows of left stream, it needs all the rows of second stream, ignoring the

partitioning of second stream. As data is divided into partitions data is being pushed around in different chunks, which is time consuming. To avoid moving data as and when required, there is an option to broadcast the smaller stream of data to every node, then comparison becomes faster. Broadcast is only possible for smaller datasets. Dataflow/Spark internally try to see if they can broadcast based on measuring/sampling the data, but sometimes different conditions like slow data ingestion from source can result in timing out of the broadcast operation. So ADF Dataflow provides options to override 'Auto' settings. Broadcast option can be set to 'None' in case broadcast is not working properly, it does not mean there will be performance impact, as broadcast would not have been successful anyway and it is just engine not able to get the right settings. Similarly for small streams users can set the broadcast option to true, like left stream, right stream, or both.

What are inline Datasets?

ADF has concept of Datasets, which are reusable entity can be used across various runtimes like copy, Dataflows, lookup etc. ADF Copy and ADF Mapping Dataflows are different engines, and there are times that not all properties or types in Datasets are compatible across Copy and Dataflows. Also, many times there is no need for reusability. In such scenarios Dataflows supports inline datasets, which define the shape and attributes of data within the Dataflow source and sinks. Some sources and sinks like CDM (Common Data Model), hive and Delta are only supported through inline Datasets. Inline datasets are more compliant with Dataflows and might provide more functionality in future as they are created using the Dataflow engine itself which will be consuming the same.

Support for CDC (Change Data Capture) in Dataflows?

Dataflows have CDC [Change data capture] support which is growing with different connectors. In the case of built in CDC support Dataflows are managing the watermark/indicators (called checkpoint) for till what point data is read. In the next run data will be read from what was read in the last run. Checkpoints are fully managed by ADF in the case of CDC. As for writing this document, we are adding support for different connectors and have support for a few connectors. There are 2 types of CDC that will be supported in Dataflows.

1. **Incremental snapshot**

   In this scenario, modified/added rows since last run are picked from the source. During and at end of the run the checkpoint is updated within ADF managed storage. This checkpoint information will be used in the next run to resume from last read data. Deletes and intermediate updates are not supported in this version of CDC. In case of files this is the only available type and supported through ‘last modified date’ attribute from files’ metadata. In the case of SQL kind of sources, it is supported through a water mark column specified by users.

2. **Full fidelity CDC**

   In this version rather than current snapshot all changes are evaluated and read. This is achieved by using “Log sequence number” (LSN) or equivalent depending on source capabilities. Dataflows will store the checkpoint during and after the run. In the next run Dataflow will use checkpoint (which contains LSN) to read the incremental changes. Full fidelity CDC is not currently available in Dataflows.

CDC can also be supported in batch mode or real time continuous mode. In the case of batch mode different triggers can be set to start the incremental load. In case of continuous mode, Dataflow will keep looking for more data batch after batch. Continuous mode is not currently available in Dataflows.

Pricing for Dataflows?

Dataflow is charged based on core hours used, which means how many cores are used for how much time. From the moment a cluster is allocated for the run and till that cluster is running job or being kept alive for next set of jobs, cores hours are being counted and charged. Apart from core hours, there are disk and egress charges. Disks are not used in each run, generally when there is more data than a cluster can handle disks are added on demand when cluster is running out of memory. Egress charges are applicable only when data is leaving Azure Data center boundary.

Does integration runtime be charged: Integration runtimes (IRs) are specification of the clusters to be created, itself IRs will not get charged. Only when there is a Dataflow activity run that will result in a cluster creation, this cluster from the time of submission of creating this cluster till this cluster is running charges will be applied.

Datetime/Date/Timestamp handling in Dataflows

Datetimes are stored in any system as a big number. While reading it the datetime is converted to a human readable string based on the local/format specified. Dataflows handle date/datetime in similar way. Date/Datetime/Timestamp is processed in Dataflows natively in an integer, it is read from sources in similar format and written to sinks in similar formats. When UX shows a Datetime it converts to the local browser scale. In case you want to read to see the date in different format you can use a Derived Column transformation with functions like toString(myDatetimeColumn, "format") to see the date in a particular format temporarily. Otherwise to a target system you would like to send the date as is without putting it in any format. While reading the datetime from a delimited text file or any other system where it is stored as string in a format, parsing the date from right format is required, this can be done in source transformation, or it can be done in Derived Column transformation by using functions like toDate(myDateColumn, "format"). After converting string to date, as mentioned earlier it becomes a number and as column type is date, UX will show it in current browser local for better readability.

Does Data flow support static IPs and SHIR?

- Dataflow does not support static IPs as rest of ADF does
- Dataflow also does not support SHIR

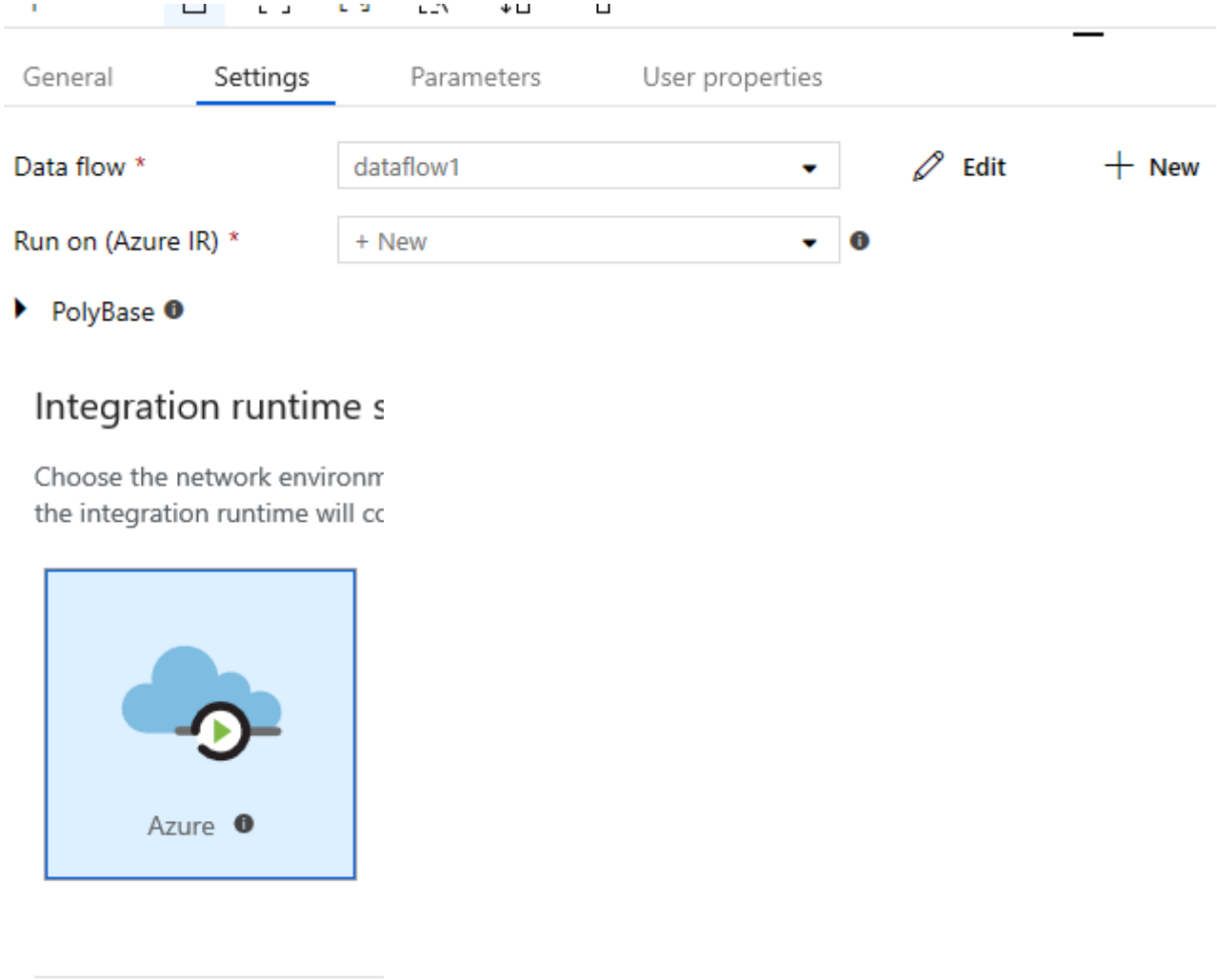
If user wants SHIR to reach behind VNet, customer can use Managed Vnet.

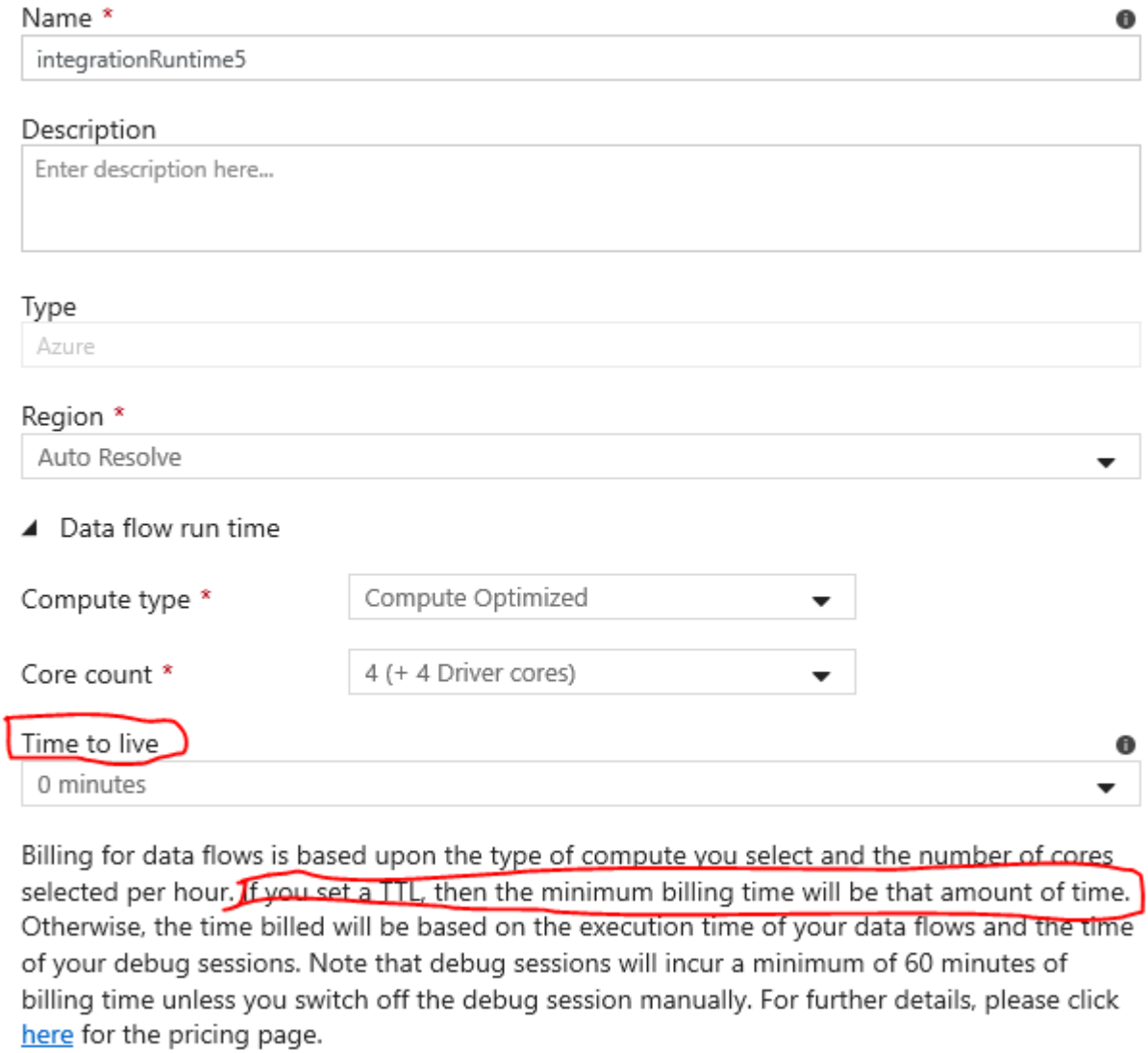
Can we share debug cluster with other team member?

A different session will be created for each factory-browser combination, so it's not possible to share cluster between different workstations or users(developers/data engineers) for dataflow in debugging mode

Can I go lower in cost than default configuration

This can be achieved to some extent by optimizing integration runtime compute. Select a custom cluster you need you create a new Integration Runtime, you can do this from the Data Flow activity in the settings tab


General
Settings
Parameters
User properties
Data flow *
dataflow1
Edit
New
Run on (Azure IR) *
+ New
PolyBase
Integration runtime s
Choose the network environm
the integration runtime will cc
Azure



Each Dataflow has different staging storage folders. Can we set the same staging storage folder for several Dataflows?

Yes, this is a supported scenario, we will be using guid for every sink.

Cheat sheet for transformations

Scenario	Transformations to use
Adding new columns	Derived Column
Duplicating columns	Select
Selecting columns	Select and Sink
Flattening array of data in rows	Flatten
Deduping data	Aggregate
Updating or deleting rows	Alter Row
Calculating moving average	Window
Looking up rows of data from another stream	Lookup or Call
Parsing JSON data in a blob column	Parse
Storing hierarchical data in Json	Stringify
Parsing Datetime	Derived Column
Separating data based on conditions	Conditional Split
Combining different data streams	Union

How good have you found this content?