PROJECT DESCRIPTION

OPTIMIZED BATCH PROCESSING USING APACHE NIFI

This project involves the incorporation of Apache NIFI, a reliable data distribution system, to batch-process large files. It assigns sequence numbers to these flowfiles and employs SQL loader to load them into an Oracle database. The project's primary objective is to enhance the efficiency of loading large files during the ingestion process. Apache NIFI is recognized for its trustworthiness in transferring and processing data across diverse systems, making it an ideal ingestion platform.

The steps required for the ETL process are:

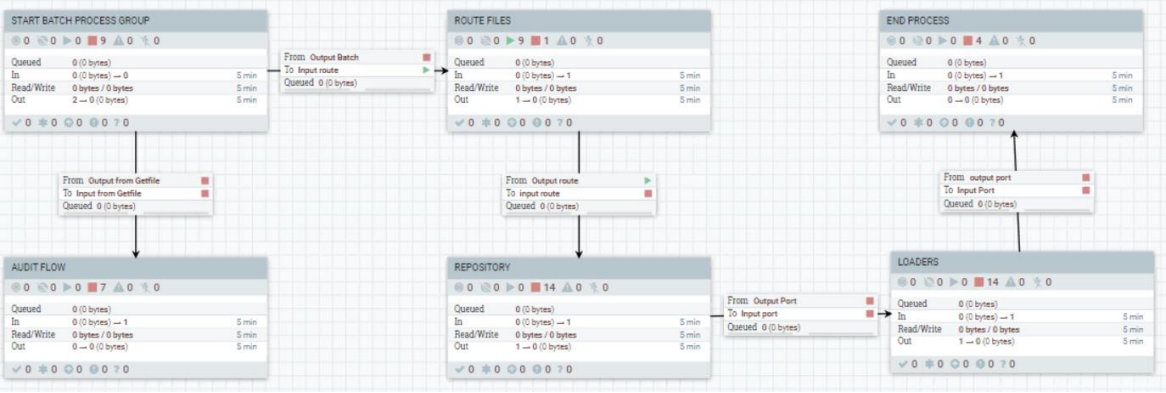
- Ingestion of large files from server into Apache Nifi.
- Generating batch sequence for N files and transforming the files.
- Loading of files in batches into Oracle DB via SQL Loader.

The Problem

The challenge encountered during this project revolved around the loading of substantial flowfiles into an Oracle database through Apache NIFI using a JDBC Connection. The issue a hand was the prolonged duration required to insert data into the database, further compounded by the adverse impact on the system's performance when attempting to process these large files all at once.

Design

The system had to be designed in such a way that files were processed in batches then loaded into the oracle database.

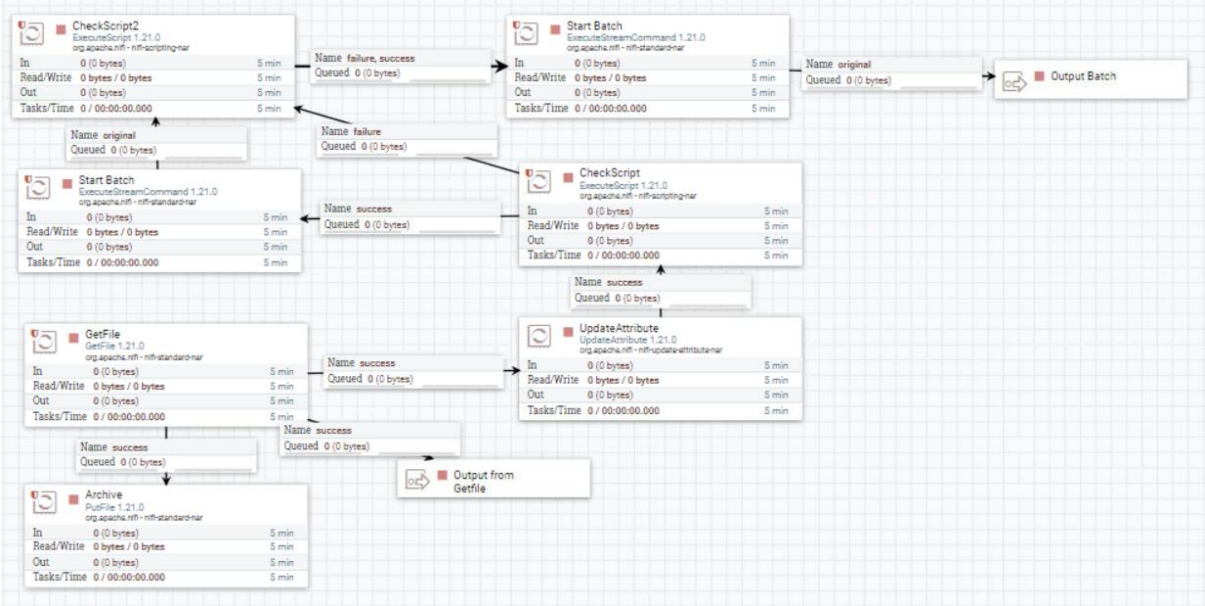


ETL BATCH PROCESS

- The Getfile automatically ingest N files into the system from the server where the files are kept.
- The flowfiles pass through the UpdateAttribute process for defined attributes to be assigned to the flowfiles.
- A batch sequence starts and generates a sequence number then attaches the number to the N flowfiles.
- They are distributed evenly by the RouteOnAttribute processor for faster processing.
- After processing the flowfiles are loaded into the database via SQL loader.
- The End batch ends the process for the next batch sequence to generate.
- Another set of flowfiles are ingested into the sequence after the process ends.

Here is the breakdown of each process group

START BATCH PROCESS GROUP



The figure above shows the initial process of the flow. The function of each processors are described below:

GetFile: This enables the flowfile to be ingested from the server into the system then it route to the updateattribute and archive processor.

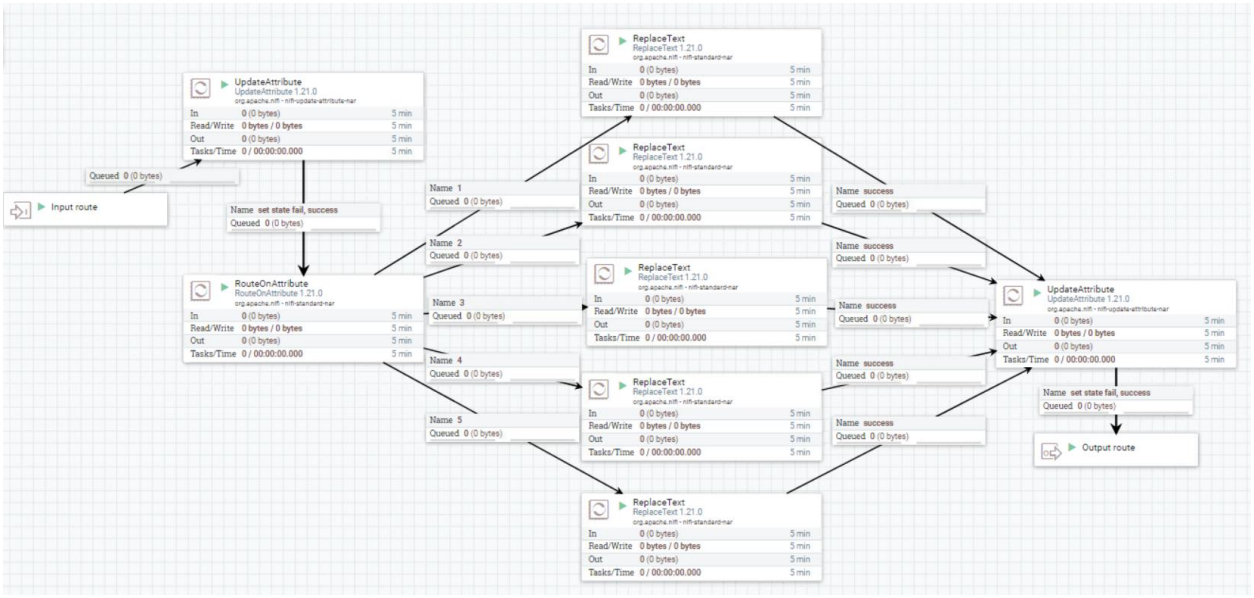
UpdateAtrribute: Attributes are allocated to the flowfiles required for processing.

ExecuteScript (CheckScript): A condition is set in the processor to check the count of flowfiles. The attribute was configured in the updateattribute processor.

ExecuteStreamCommand (StartBatch): A batch sequence is generated and group for a set of flowfiles.

PutFile (Archive): It stores flowfiles for record purpose.

ROUTE FILES



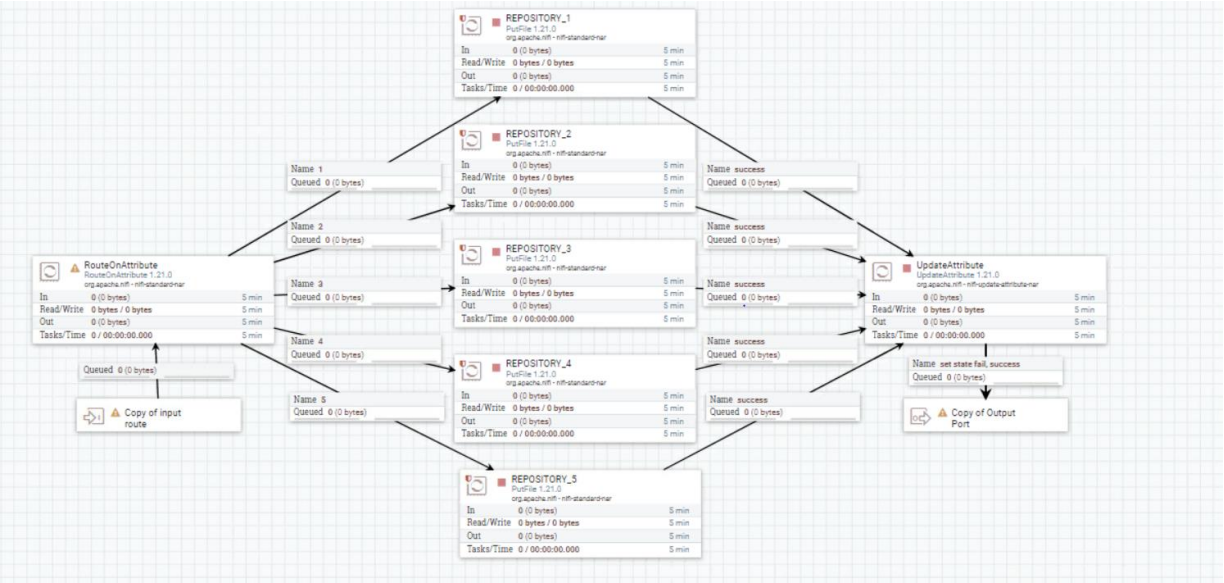
The figure above shows the second process of the flow. The function of each processors are described below:

UpdateAtrribute: Attributes are allocated to the flowfiles required for processing.

RouteOnAttribute: Route files to N number of Replacetext Processors for transformation.

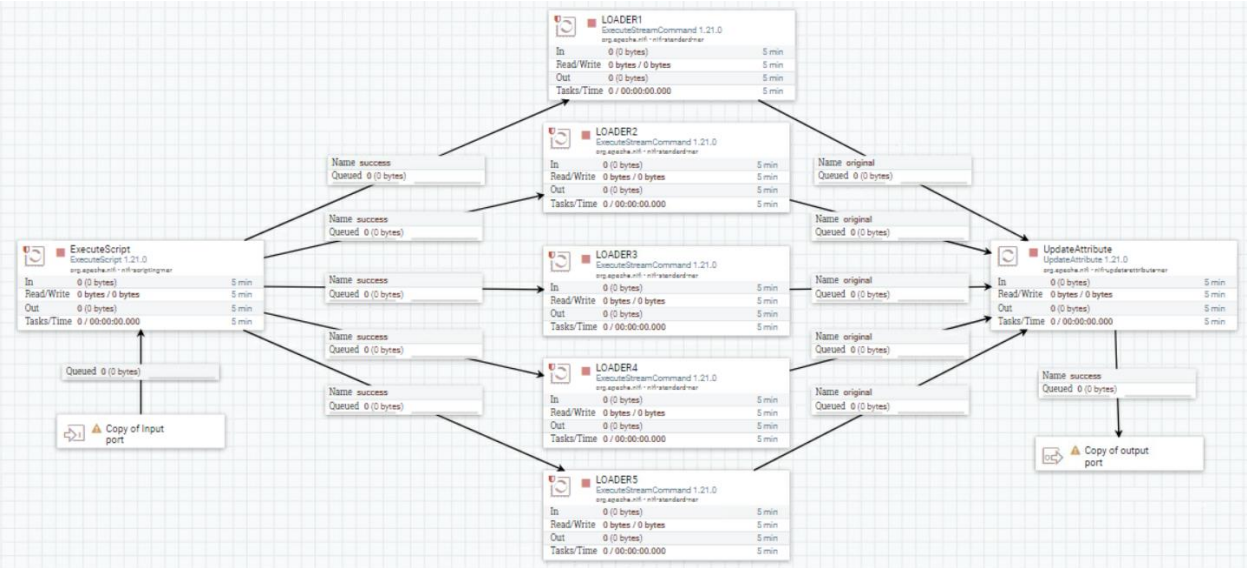
ReplaceText: Transformation of the flowfiles occurs in this processor.

REPOSITORY



The figure above shows the repository where the flowfiles are stored in a file path after processing.

LOADERS

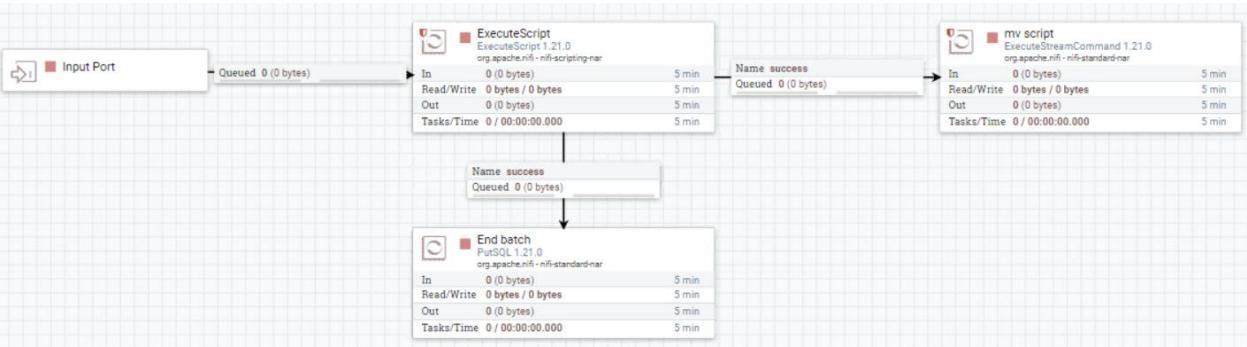


ExecuteScript: A groovy script is used in this processor to route the number of flowfiles as defined in the script.

ExecuteStreamCommand (Loader): It executes the shell script that loads the data into a table via SQL loader.

UpdateAttribute: Attributes are allocated to the flowfiles.

END PROCESS



ExecuteScript: A groovy script is used in this processor to route a flowfile to the executestreamcommand processor.

***ExecuteStreamCommand (mv script)*:** At this point the process ends and new set of files are pushed to the directory for the next processing of files.

PutSQL (End Batch): It ends the batch sequence that was generated.

Conclusion

Enhancing the database loading performance can be achieved by implementing SQL loader as a viable alternative to JDBC connections. By leveraging the capabilities of SQL loader within the Apache NIFI interface, the ingestion and processing of flowfiles in a batch sequence can significantly expedite the overall data processing and loading, thereby optimizing system efficiency and performance.