HITACHI
Inspire the Next

Pentaho and Big Data Ingestion Patterns

This page intentionally left blank.

Contents

Overview 1
- Before You Begin 1
- Use Case: Many Small Text Files 1

Data Ingestion to Big Data 2
- Data from External JDBC Databases 3
  - Use Sqoop to Ingest Data 3
  - Use Pentaho JDBC to HDFS 4
  - Use PMR JDBC to HDFS 4
- Data from Local/External File Sources 5
  - Use Pentaho File to HDFS 5
  - Use PMR File to HDFS 6
- Data from Realtime Systems 7
  - Use Kafka or JMS Sources 7
  - Use Kafka or MQTT Sources 8

Data Processing in Big Data 9
- Text-Based Data in Large Files 10
  - Processing Large Files from HDFS with PMR 10
  - Processing Large Files from HDFS with AEL Spark 10
- Text-Based Data in Many Small Files 11

Related Information 12

This page intentionally left blank.

Overview

This document covers some best practices on collecting Pentaho Data Integration (PDI) and Hadoop usage patterns for data ingestion and processing.

Our intended audience is Pentaho administrators, or anyone with a background in Hadoop and data ingestion who is interested in choosing the best methods for processing data.

The intention of this document is to speak about topics generally; however, these are the specific versions covered here:

Software	Version(s)
Pentaho	7.x, 8.0

The Components Reference in Pentaho Documentation has a complete list of supported software and hardware.

Before You Begin

Before beginning, use the following information to prepare for the procedures described in the main section of the document.

This document assumes that you have knowledge of Pentaho and Hadoop and already know what format your data will arrive in.

Use Case: Many Small Text Files

Janice has many small files of product descriptions in Excel spreadsheets. She needs to load this data as quickly as possible so that Marketing can redo the product catalog.

Janice decides to use a data loading pattern of a distributed load process, with the files processed in parallel by different Hadoop nodes.

Data Ingestion to Big Data

Data ingestion is the process of getting data from external sources into big data. There are different patterns that can be used to load data to Hadoop using PDI. The preferred ingestion format for landing data from Hadoop is Avro.

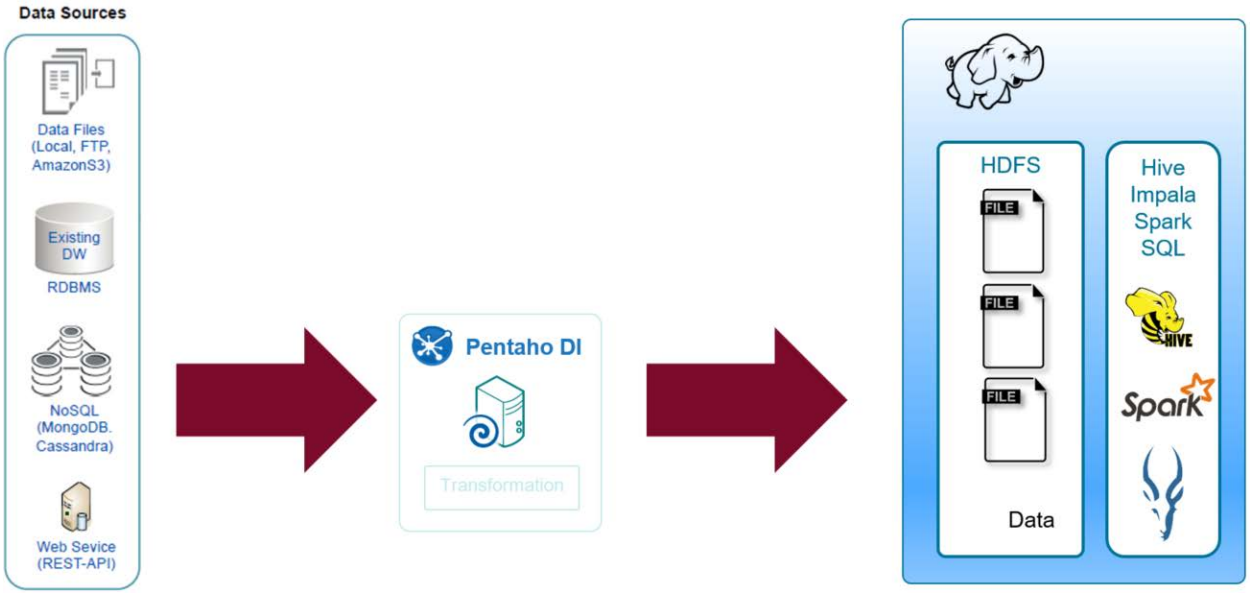

Figure 1: Ingestion to Big Data

You can find details on these topics in the following sections, sorted by where your data is arriving from:

- Data from External JDBC Databases
- Data from Local/External File Sources
- Data from Realtime Systems

Data from External JDBC Databases

When you collect source data in external Java Database Connectivity (JDBC) databases, you have multiple methods you can choose from to ingest data:

Table 1: Ingestion from External JDBC Databases

Situation	Available Method(s)
Data is accessible from any Hadoop node. Nodes have connectivity to the external database. No transformation/change is needed.	Use Sqoop to ingest data.
Data is accessible from any Hadoop node. Nodes have connectivity to the external database. Data may or may not need transformation.	Use Pentaho MapReduce (PMR) JDBC to Hadoop Distributed File System (HDFS). **OR** Use PDI Cluster on YARN.
Data is not accessible from Hadoop nodes. Data is only accessible by edge nodes or Pentaho Server.	Use Pentaho JDBC to HDFS. **OR** Use Pentaho distributed processing (PDI Cluster).

Use Sqoop to Ingest Data

Using Sqoop to ingest your data involves a distributed load process, with multiple connections to the source database from different Hadoop nodes.

The HDFS destination for your data can be text, Avro, or Parquet. Avro schemas are created automatically by Sqoop based on source data. You can also enable different Avro or Parquet options if you wish.

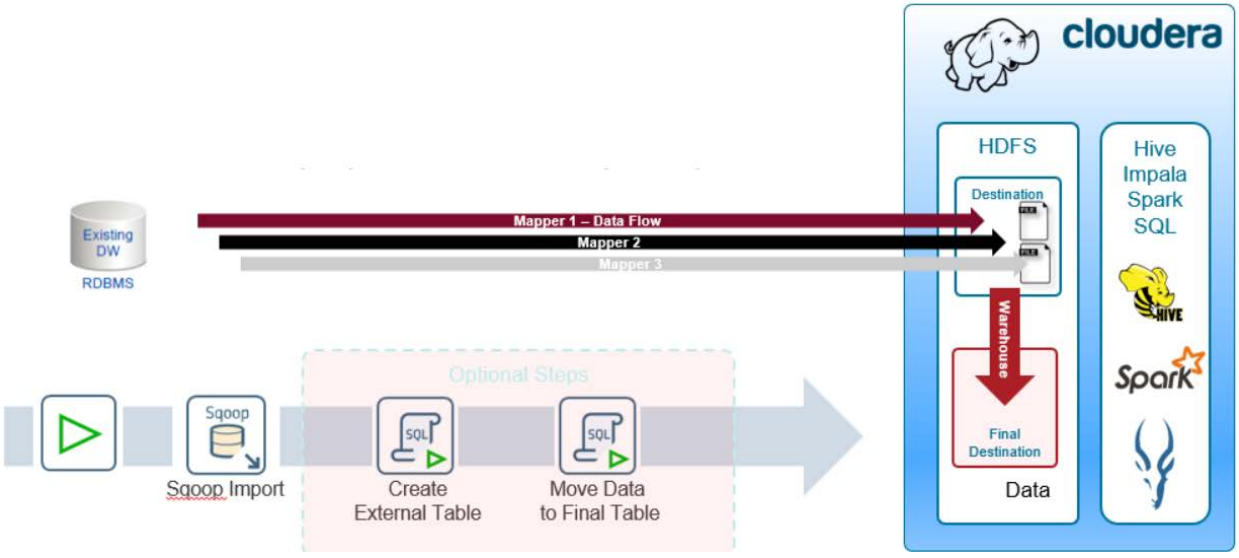

Figure 2: Using Sqoop to Ingest Data

Use Pentaho JDBC to HDFS

In this case, you can have a single worker process the entire dataflow using PDI (Kitchen/Pan/Carte). The HDFS destination for your text can be text, Parquet, or Avro (plain Avro). You can also enable different Avro or Parquet options if you wish.

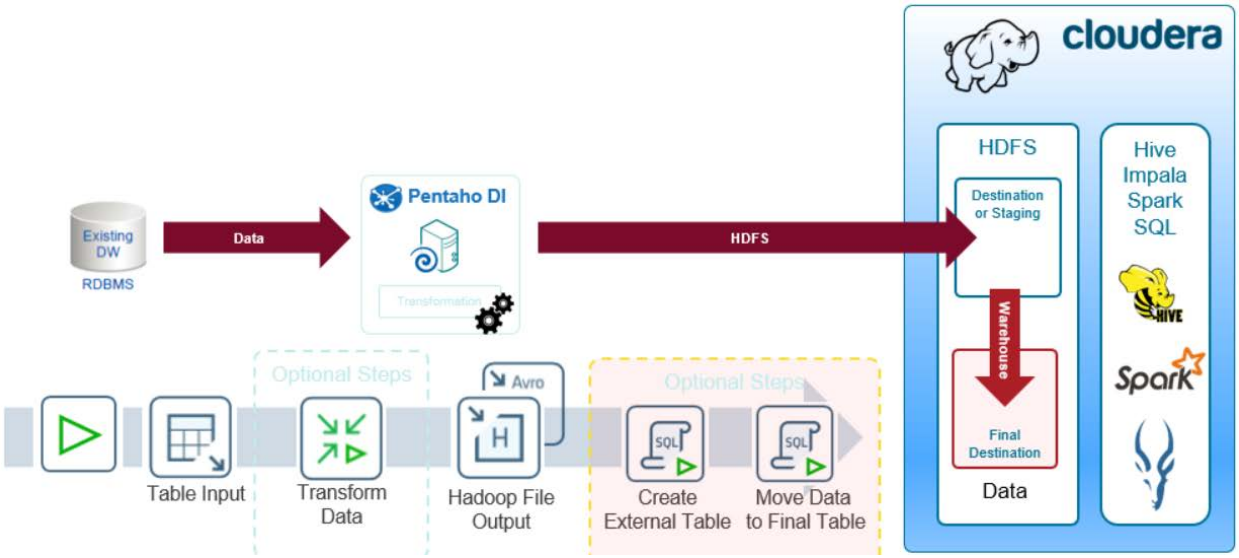

Figure 3: Use Pentaho JDBC to HDFS

Use PMR JDBC to HDFS

This method uses a distributed load process, with multiple connections to source databases from different Hadoop nodes. You must define how to segment the data ranges. The HDFS destination for your text can be text, Parquet, or Avro. You can also enable different Avro or Parquet options if you wish.

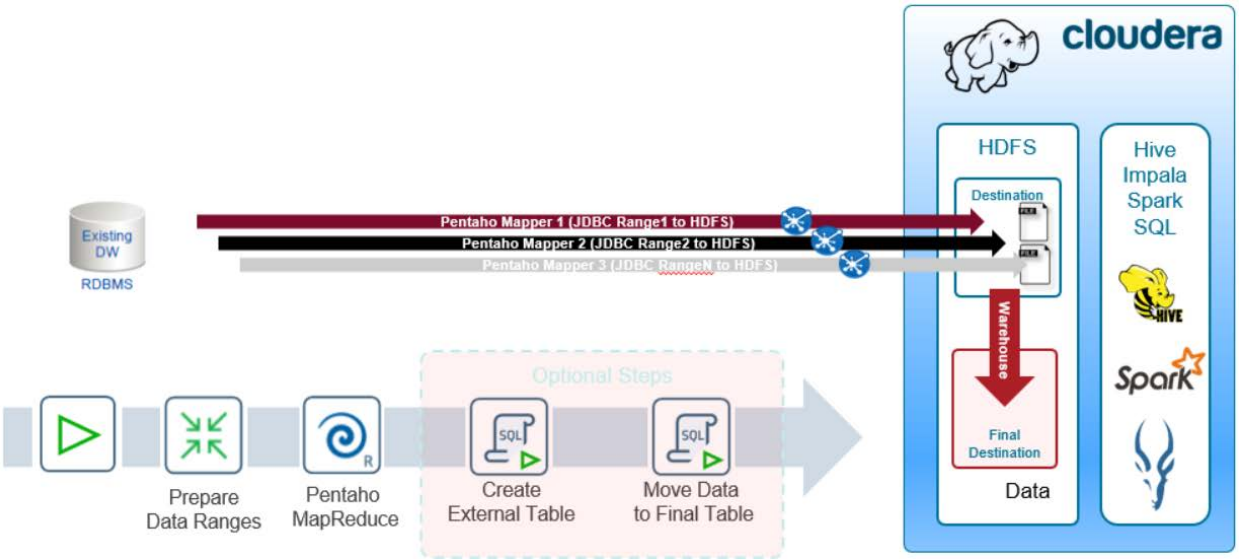

Figure 4: Use PMR JDBC to HDFS

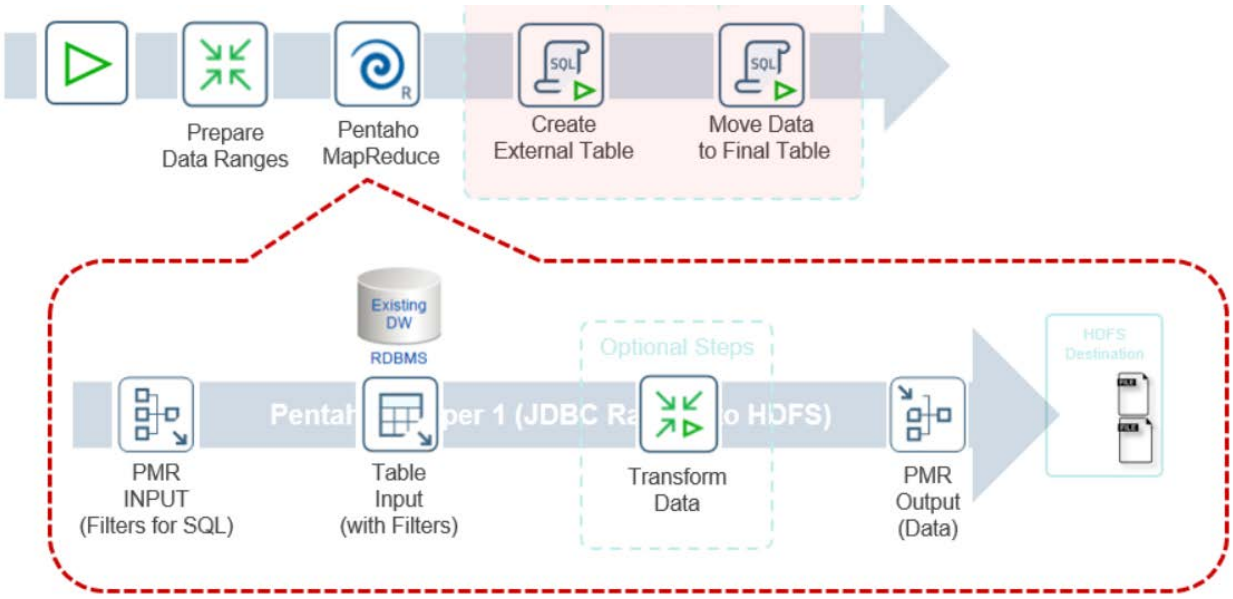

Figure 5: PMR Breakdown with Optional Steps

Data from Local/External File Sources

When you store source data in local/external filesystems, you may find one of these patterns useful:

Table 2: Ingestion from Local/External File Sources

Situation	Available Method(s)
Data is not accessible from Hadoop nodes. Data is only accessible by edge nodes or Pentaho Server.	Use Pentaho File to HDFS.
Data is accessible from any Hadoop node. Nodes have connectivity to the external database.	Use PMR File to HDFS.

Use Pentaho File to HDFS

In this case, you can have a single worker process the entire dataflow using PDI (Kitchen/Pan/Carte). The files are processed sequentially.

The HDFS destination for your text can be text, Parquet, or Avro (plain Avro). You can also enable different Avro or Parquet options if you wish.

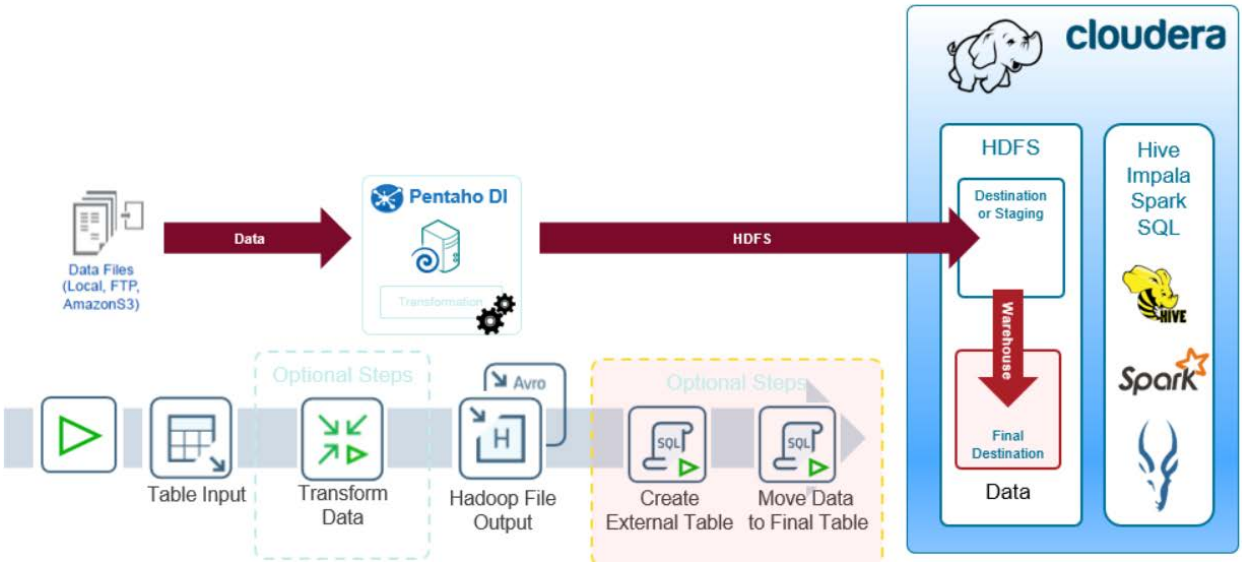

Figure 6: Pentaho File to HDFS

Use PMR File to HDFS

This method uses a distributed load process to distribute the file loading in parallel. The set of files is processed by different Hadoop nodes.

The HDFS destination for your text can be text, Parquet, or Avro. You can also enable different Avro or Parquet options if you wish.

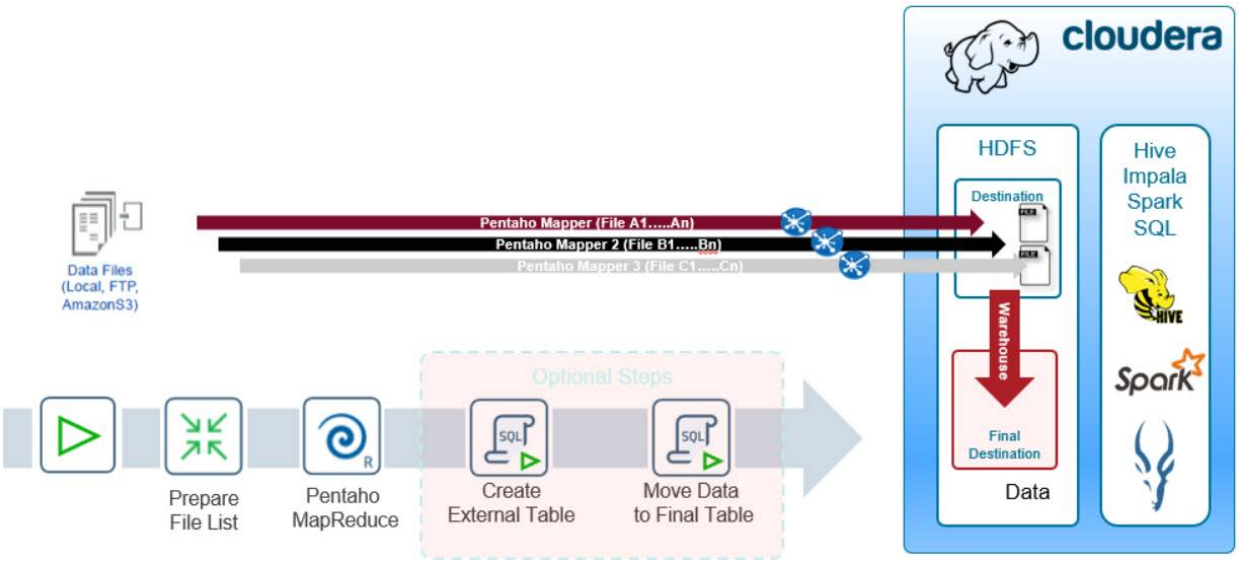

Figure 7: PMR File to HDFS

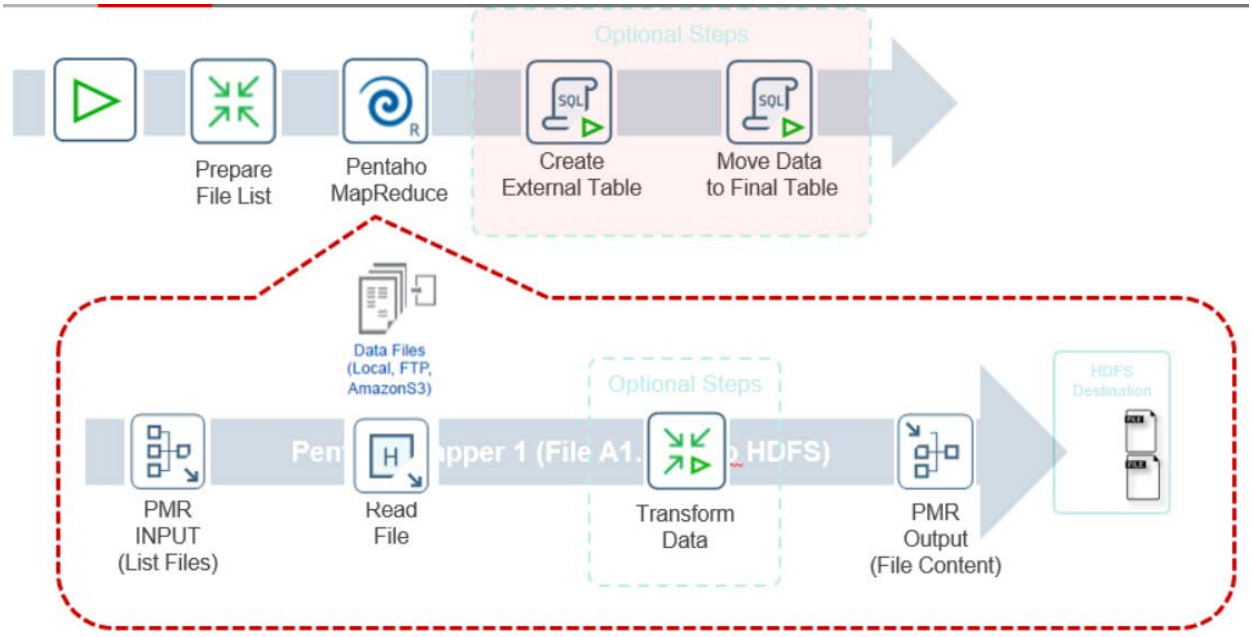

Figure 8: PMR Breakdown with Optional Steps

Data from Realtime Systems

In cases where your data comes from realtime systems like messages or streaming sources, you have these pattern options available to you:

Table 3: Ingestion from Realtime Systems

Data is...	Available Method(s)
Event-driven	Use Kafka or Java Message Service (JMS) sources.
Microbatch-driven	Use Kafka or Message Queue Telemetry Transport (MQTT) sources.

Use Kafka or JMS Sources

When you have event-driven data coming in, this pattern gives you an event handler to receive the request and fire independent processes to process the request.

You can then use any of the patterns described in Realtime Data Processing with PDI to process the data.

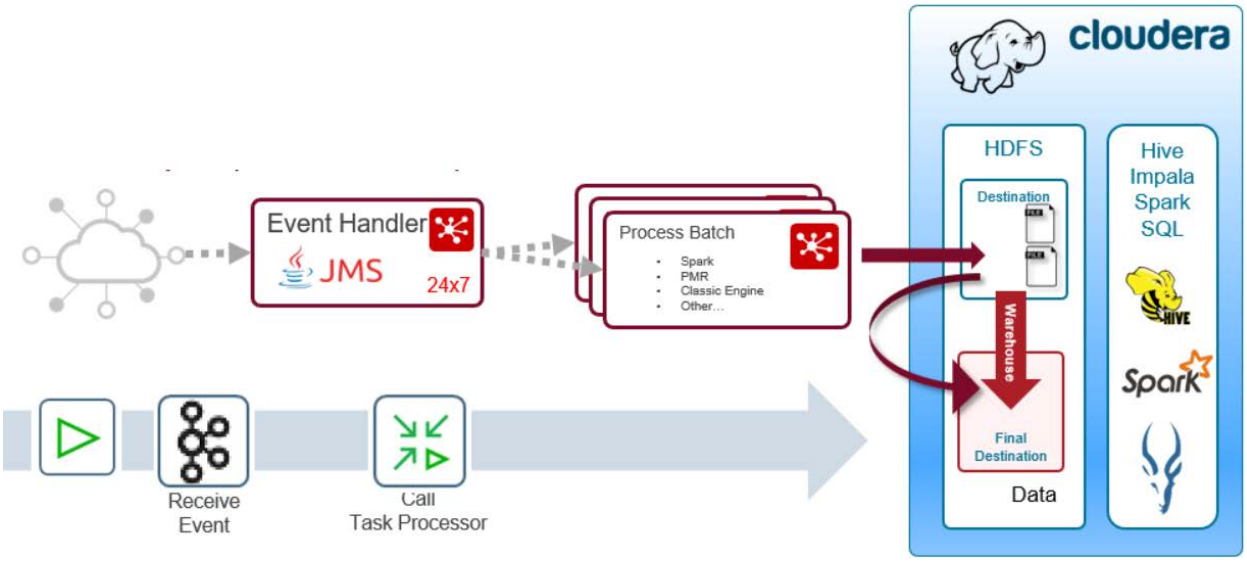

Figure 9: Use Kafka or JMS for Event-Driven Data

Use Kafka or MQTT Sources

When you have microbatch-driven data coming in, this pattern gives you a data collector that will receive your data in real time and store it in your desired location, with no processing. After a set threshold is reached (whether that is a number of records, a window of time, or a chunk identifier), the data collector fires an event. At that point, use the previously described pattern, Use Kafka or JMS, to process the data.

This pattern has the advantage of scalability, since you can add more event handlers and microbatch your parallel processing.

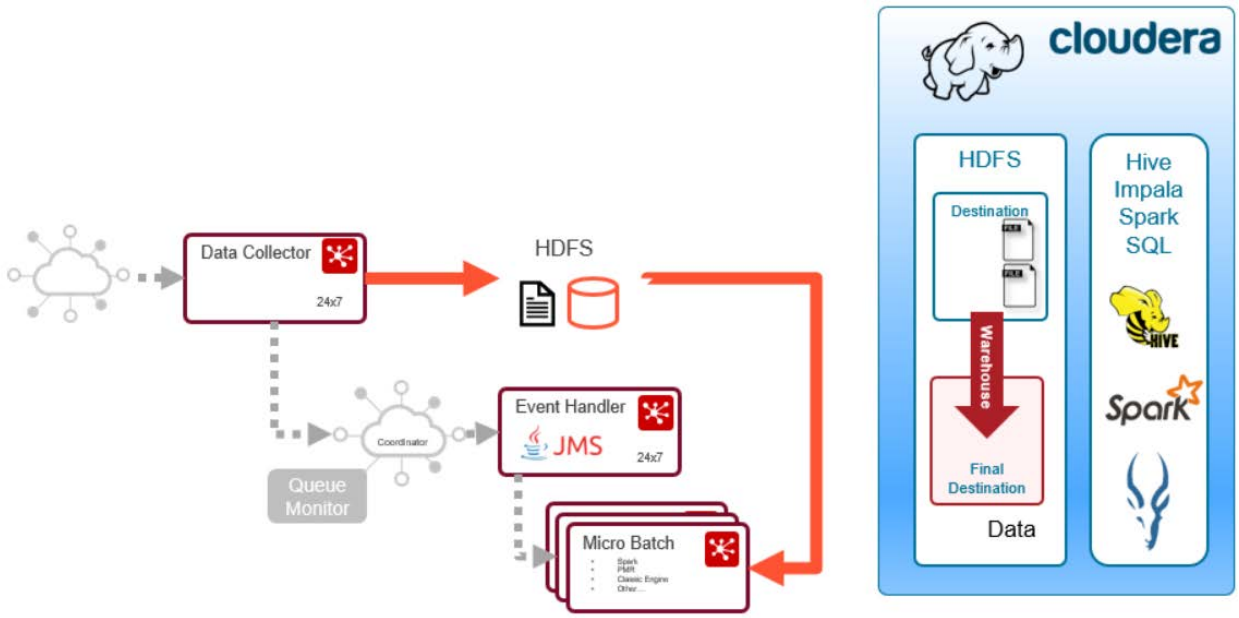

Figure 10: Use Kafka or MQTT for Microbatch-Driven Data

Data Processing in Big Data

Several patterns are available for processing data stored in Hadoop that is accessible via HDFS, Hive, Impala, and so forth.

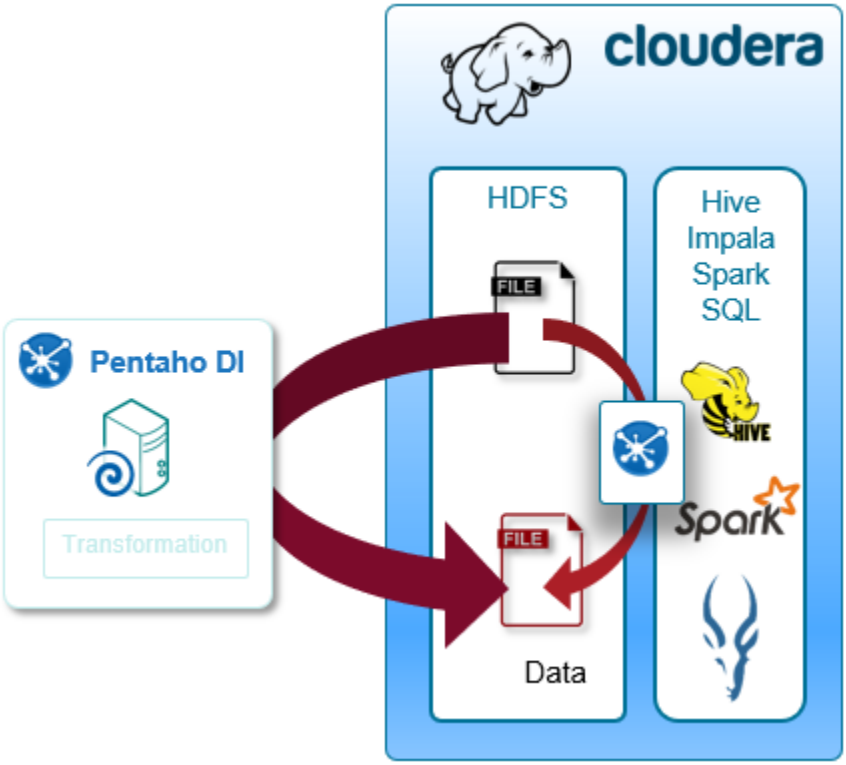

Figure 11: Data Processing

Input and output format may vary from case to case, and different patterns apply best to each case. For regular data processing, you may want to clean, transform, and/or extend your data. For staging your data, you may want to do things like create external Hive definitions on top of your staging data, or move data from staging to a final destination, with proper partitioning and file formatting.

When you need to transform, clean, classify, extend, or filter data that is stored in HDFS, you need to maximize data locality usage.

You can find details on these topics in the following sections:

- Text-Based Data in Large Files
- Text-Based Data in Many Small Files

Text-Based Data in Large Files

When your text-based input is stored in large files like XML, JSON, or text, patterns that may work for you are to process with PMR or with Adaptive Execution Layer (AEL) Spark.

Processing Large Files from HDFS with PMR

This processing pattern uses data locality. The source can be text, XML, JSON, or Avro. The HDFS destination for your data can be text, Parquet, or Avro. You can also enable different Avro or Parquet options if you wish.

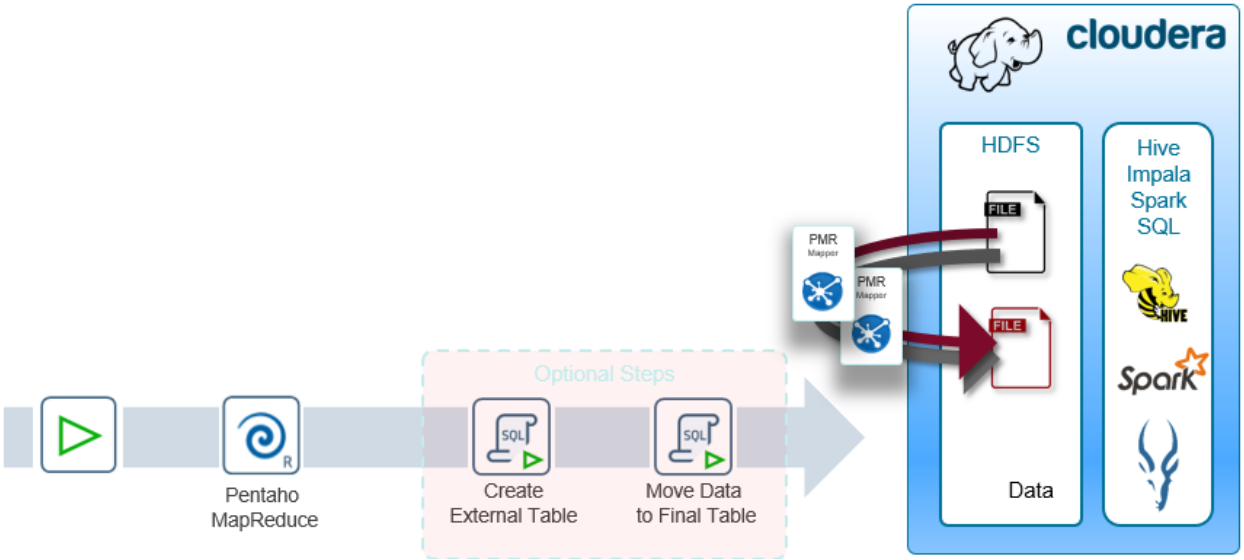

Figure 12: Processing Large Files with PMR

Processing Large Files from HDFS with AEL Spark

This processing pattern uses data locality. The source can be text, XML, or JSON. The HDFS destination for your data will be text.

Figure 13: Processing Large Files with AEL Spark

Text-Based Data in Many Small Files

This pattern uses a distributed load process to distribute the file loading in parallel. The set of files is processed by different Hadoop nodes. This method does not use data locality.

The data source can be text, XLS/XLSX, or Avro, with other formats supported by Pentaho input. The destination can be text, Parquet, or Avro. You can also enable different Avro or Parquet options if you wish.

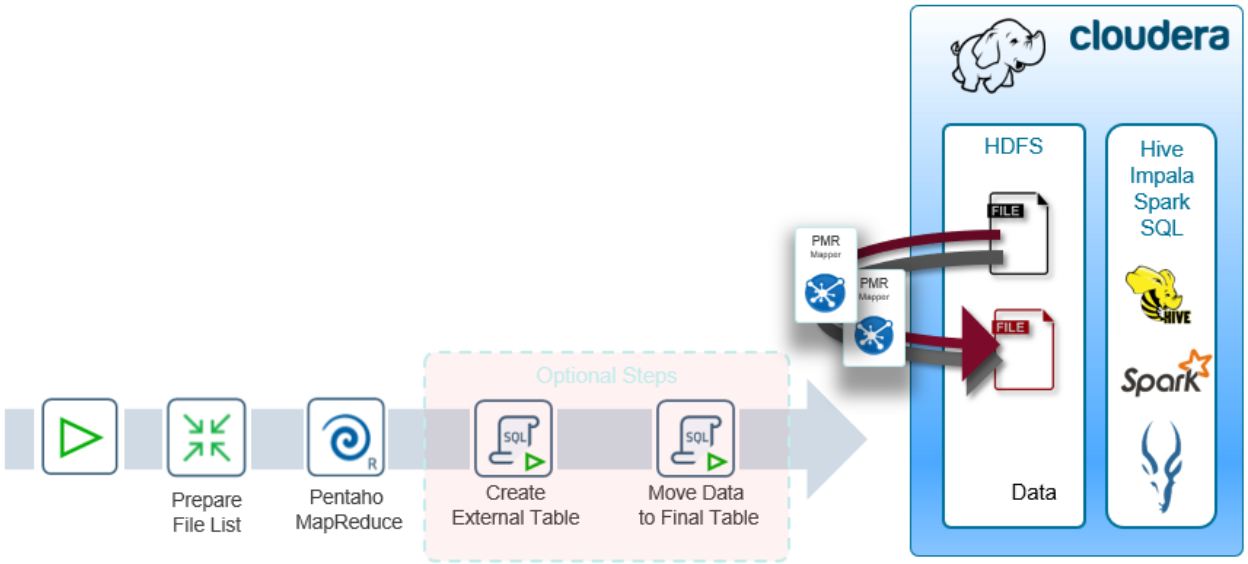

Figure 14: Processing Small Files Using Distributed Load Process

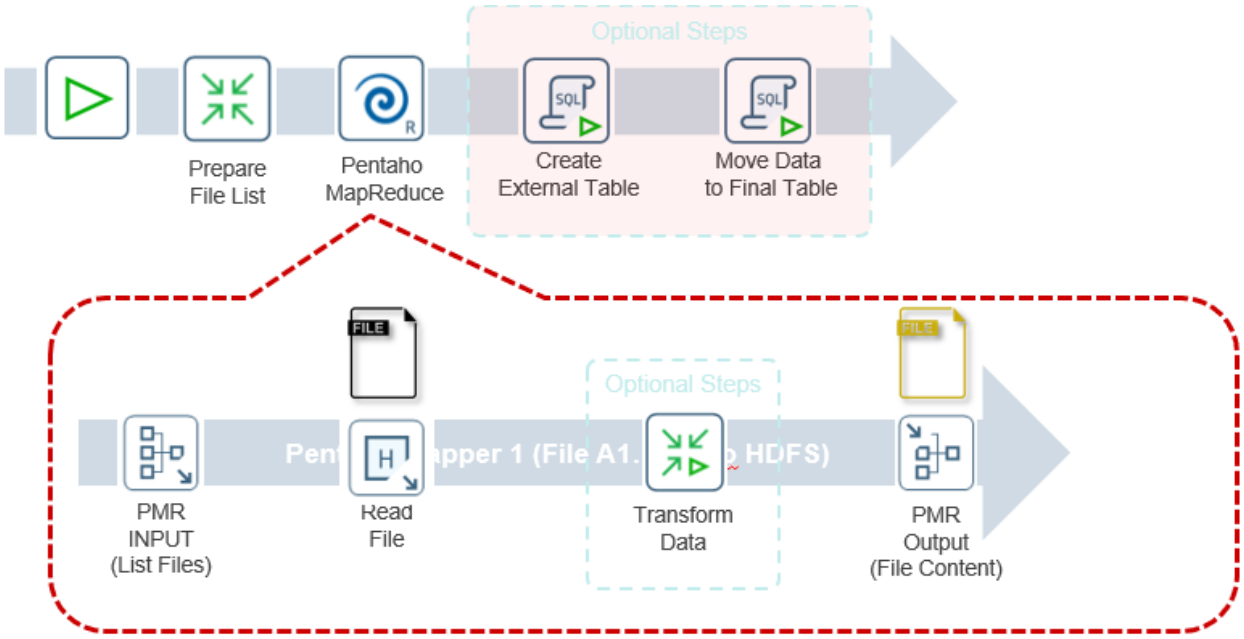

Figure 15: PMR Breakdown with Optional Steps

Related Information

Here are some links to information that you may find helpful while using this best practices document:

- Pentaho Big Data On-Cluster Processing
- Pentaho Components Reference
- Set Up Pentaho to Connect to an Amazon EMR Cluster
- Start a PDI Cluster on YARN